Cribl®

Cribl Stream Documentation Manual

Version: v4.6

Generated: 2024-04-17 06:56:20

1. Introduction

1.1. About Cribl Stream

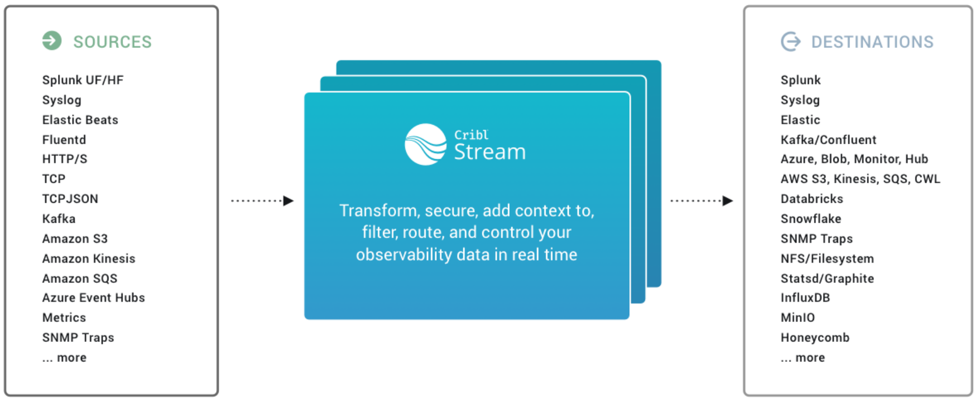
All the docs to goat you started with Cribl Stream

- Download all docs as a PDF – v.4.5.1 | Download Cribl Stream | Get Cribl.Cloud
- Questions not answered here? We'd love to help you. Meet us in #Cribl Community Slack – sign up here.

What Is Cribl Stream?

Cribl Stream helps you process machine data – logs, instrumentation data, application data, metrics, etc. – in real time, and deliver them to your analysis platform of choice. It allows you to:

- Add context to your data, by enriching it with information from external data sources.
- Help secure your data, by redacting, obfuscating, or encrypting sensitive fields.
- Optimize your data, per your performance and cost requirements.



Sources, Cribl Stream, Destinations

Cribl Stream ships in a single, no-dependencies package. It provides a refreshing and modern interface for working with and transforming your data. It scales with – and works inline with – your existing infrastructure, and is transparent to your applications.

Who Is Cribl Stream For?

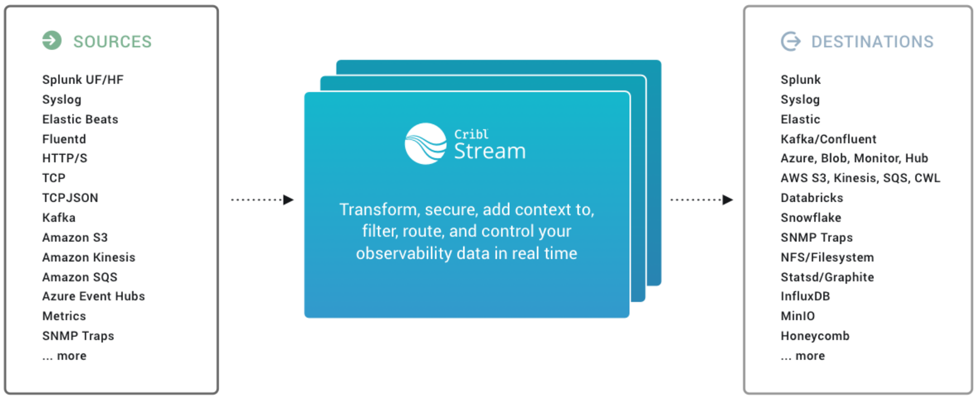
Cribl Stream is built for administrators, managers, and users of operational/DevOps and security intelligence products and services.

1.2. Basic Concepts

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

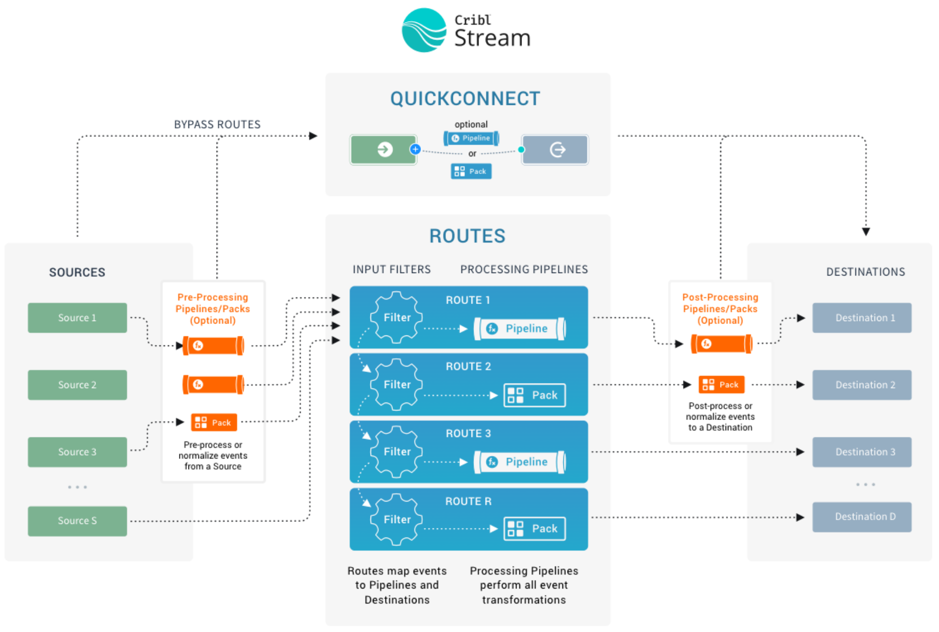
---

As we describe features and concepts, it helps to have a mental model of Cribl Stream as a system that receives events from various sources, processes them, and then sends them to one or more destinations.



Sources, Cribl Stream, Destinations

Let's zoom in on the center of the above diagram, to take a closer look at the processing and transformation options that Cribl Stream provides internally. 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

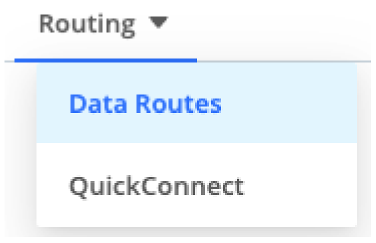
Sources are configurations that enable Cribl Stream to receive data from remote senders (Splunk, TCP, Syslog, etc.), or to collect data from remote file stores or the local machine.

QuickConnect

QuickConnect is a graphical interface for setting up data flow through your Cribl Stream 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

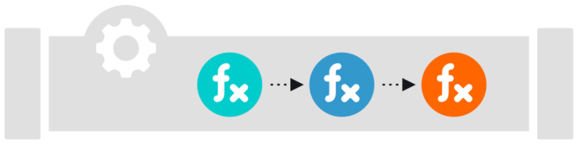
Routes evaluate incoming events against **filter** expressions to find the appropriate Pipeline to send them to. Routes are **evaluated in order**. Each Route can be associated **with only one** Pipeline and one output (configured as a Cribl Stream **Destination**).

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.

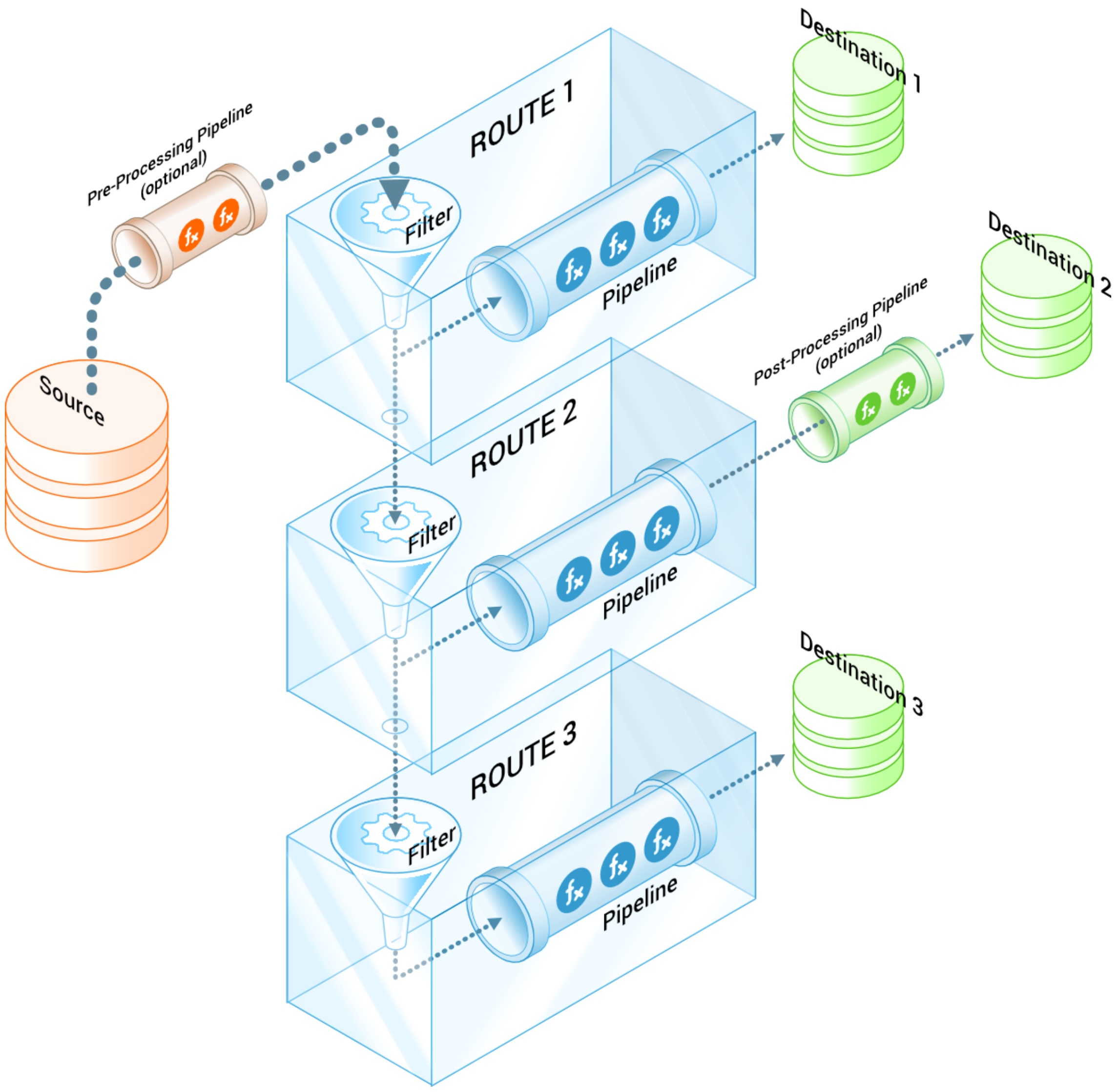
However, if you disable the Route's `Final` flag, one or more event **clones** will be sent down the associated Pipeline, while the original event continues down Cribl Stream's Routing table to be evaluated against other configured Routes. This is very useful in cases where the same set of events needs 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. Events are delivered to the beginning of a Pipeline by a Route, 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 Cribl Stream Sources, and attach **post-processing Pipelines** to Cribl Stream Destinations. All Pipelines are configured through the same UI – these three designations are determined only by a Pipeline's placement in Cribl Stream's data flow.

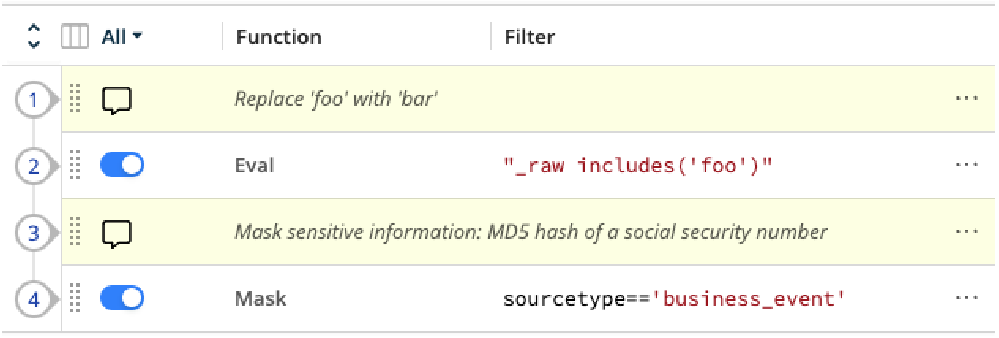


Pipelines categorized by position

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`.



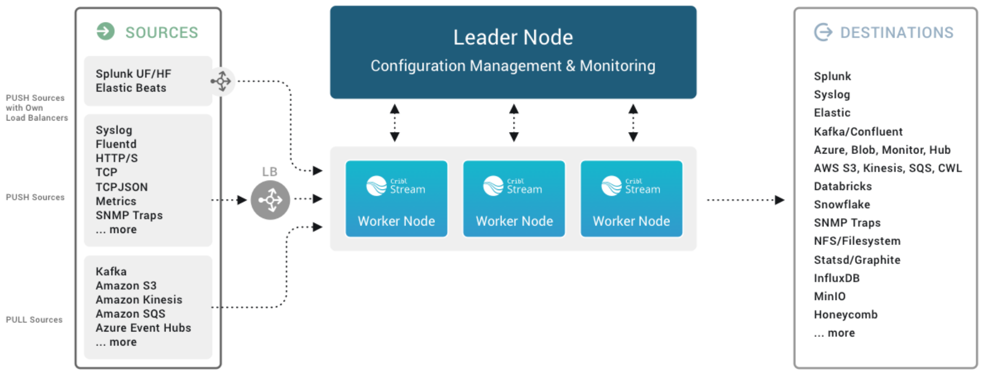
	All	Function	Filter
1		*Replace 'foo' with 'bar'*	
2		Eval	`"_raw includes('foo')"`
3		*Mask sensitive information: MD5 hash of a social security number*	
4		Mask	`sourcetype=='business_event'`

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.

A Scalable Model

You can scale Cribl Stream up to meet enterprise needs in a distributed deployment. Here, multiple Cribl Stream Workers (instances) share the processing load. But as you can see in the preview schematic below, even complex deployments follow the same basic model outlined above.



Distributed deployment architecture

1.3. Getting Started Guide

This guide walks you through planning, (optionally) installing, and configuring a basic deployment of Cribl Stream. You'll capture some realistic sample log data, and then use Cribl Stream's built-in Functions to redact, parse, refine, and shrink the data.

By the end of this guide, you'll have assembled all of Cribl Stream's basic building blocks: a Source, Route, Pipeline, several Functions, and a Destination. You can complete this tutorial using Cribl Stream's included sample data, without connections to – or licenses on – any inbound or outbound services.

Assuming a cold start (from first-time setup of a Cribl.Cloud or self-hosted instance), this guide might take about an hour. But you can work through it in chunks, and Cribl Stream will persist your work between sessions.

> If you've already launched a Cribl Stream instance (either Cribl.Cloud or self-hosted), skip ahead to Get Data Flowing.
>
> If you'd prefer a similar tutorial based on Cribl Stream's simplified QuickConnect visual UI, see our CrowdStream setup guide. All the steps shown in that tutorial work equally well in Cribl Stream.
>
> Once you've mastered all the techniques in this tutorial, check out our Distributed Quick Start, where you'll see how to apply them to a typical production deployment across multiple hosts.

Cloud or Self-Hosted?

To quote Robert Frost and Robert Plant (of Led Zeppelin), there are two paths that you can go by:

- Do this tutorial with a free Cribl.Cloud instance, hosted by Cribl. Follow the registration instructions in the section just below. This skips you past all the requirements and installation sections below – you'll have nothing to install or host. Because Cribl.Cloud always runs in distributed mode, this will require a few extra clicks (all clearly labeled) later in the tutorial's body. But it will give you immediate experience with Cribl Stream's typical production mode.
- Do this tutorial by downloading and installing Cribl Stream software on your own hardware or virtual machine. Follow the Requirements and Download and Install instructions below. You'll need to provide your own infrastructure. But if you're planning to use self-hosted/on-prem Cribl Stream in production, this will walk you through its realistic setup.

  > To fully experience a real-world, self-hosted production setup, you can switch your Cribl Stream installation to distributed mode. And then chase this tutorial with our Distributed Quick Start,

> which fully exercises distributed mode's multiple Workers and Worker Groups.

Quick Start with a Cloud Instance

As indicated just above, you can skip installing Cribl Stream software – and skip this tutorial's next several sections – by registering a free Cribl.Cloud instance. Cribl will quickly spin up a fully functioning copy of Cribl Stream for you, and manage it on your behalf. To use this fastest option:

1. In the Cribl.Cloud Launch Guide, follow the four steps listed in Registering a Cribl.Cloud Portal.
2. In the same Cribl.Cloud Launch Guide, jump ahead to follow the four steps listed in Managing Cribl.Cloud.
3. That's it! Come right back to this tutorial, and skip ahead to Get Data Flowing.

> Really, skip all the other sections of the linked Cloud Guide, and all this tutorial's sections between here and Get Data Flowing. We told you this was a quick start!

Requirements for Self-Hosted Cribl Stream

The minimum requirements for running this tutorial are the same as for a Cribl Stream production single-instance deployment.

OS

- Linux kernel >= 3.10 and glibc >= 2.17.

Example distributions: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2.

Tested so far on Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2.

System

- 1 GHz (or faster), 64-bit CPU.
- 4+ physical cores, 8 GB+ RAM – all beyond your basic OS/VM requirements.
- 5 GB free disk space (more if persistent queuing is enabled).

> We assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs) on Intel/Xeon or AMD processors; and to 1 (higher-throughput) vCPU on Graviton2/ARM64 processors.

Browser

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

Network Ports

By default, Cribl Stream listens on the following ports:

Component	Default Port
UI default	`9000`
HTTP Inbound, default	`10080`
User options	+ Other data ports as required.

You can override these defaults as needed.

Plan for Production

For higher processing volumes, users typically enable Cribl Stream's Distributed Deployment option. While beyond the scope of this tutorial, that option has a few additional requirements, which we list here for planning purposes:

- Port `4200` must be available on the Leader Node for Workers' communications.
- Git (1.8.3.1 or higher) must be installed on the Leader Node, to manage configuration changes.

See Sizing and Scaling for further details about configuring Cribl Stream to handle large data streams.

Download and Install Cribl Stream

To avoid permissions errors, you should both install and run (next section) Cribl Stream as the same Linux user. For details on creating a new user (addressing both systemd and initd distro's), see Enabling Start on Boot.

Download the latest version of Cribl Stream at https://cribl.io/download/.

Un-tar the resulting `.tgz` file in a directory of your choice (e.g., `/opt/`). Here's general syntax, then a specific example:

```
tar xvzf cribl-<version>-<build>-<arch>.tgz
tar xvzf cribl-3.5.0-fa5eb040-linux-x64.tgz
```

You'll now have Cribl Stream installed in a `cribl` subdirectory, by default: `/opt/cribl/`. We'll refer to this `cribl` subdirectory throughout this documentation as `$CRIBL_HOME`.

Run Cribl Stream

In your terminal, switch to the `$CRIBL_HOME/bin` directory (e.g,: `/opt/cribl/bin`). Here, you can start, stop, and verify the Cribl Stream server using these basic `./cribl` CLI commands:

- **Start**: `./cribl start`
- **Stop**: `./cribl stop`
- **Get status**: `./cribl status`

For other available commands, see CLI Reference.

Next, in your browser, open `http://<hostname>:9000` (e.g., `http://localhost:9000`) and log in with default credentials (`admin`, `admin`).

Register your copy of Cribl Stream when prompted.

After registering, you'll be prompted to change the default password.

That's it!

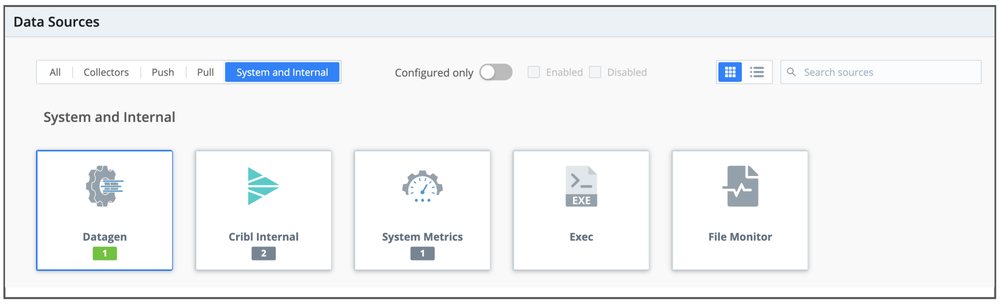
Get Data Flowing

With Cribl Stream now running – either in Cribl.Cloud or in your self-hosted copy – you're ready to configure a working Cribl Stream deployment. You'll set up a Source, Destination, Pipeline, and Route, and will assemble several built-in Functions to refine sample log data.

Add a Source

Each Cribl Stream Source represents a data input. Options include Splunk, Elastic Beats, Kinesis, Kafka, syslog, HTTP, TCP JSON, and others.

For this tutorial, we'll enable a Cribl Stream built-in datagen (i.e., data generator) that generates a stream of realistic sample log data.



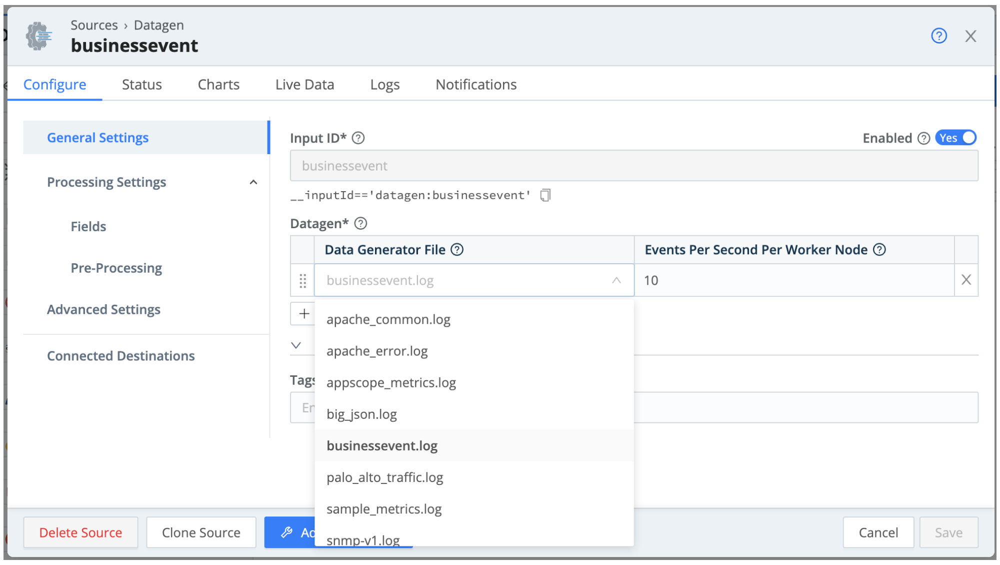
Adding a datagen Source

If you're on Cribl.Cloud or any other distributed mode, first click the top nav's **Manage** tab to select the `default` (or another) Worker Group.

1. From Cribl Stream's **Manage** submenu, select **Data** > **Sources**.
2. From the **Data Sources** page's tiles or left menu, select **Datagen**.
   (You can use the search box to jump to the **Datagen** tile.)
3. Click **Add Source** to open the **New Source** modal.
4. In the **Input ID** field, name this Source `businessevent`.
5. In the **Data Generator File** drop-down, select `businessevent.log`.
   This generates...log events for a business scenario. We'll look at their structure shortly, in Capture and Filter Sample Data.
6. Click **Save**.

If you're on Cribl.Cloud or any other distributed mode, click **Commit & Deploy** at Cribl Stream's upper right before proceeding. Then, in the resulting dialog box, click **Commit & Deploy** to confirm. You'll see a **Commit successful** message.

The `Yes` toggle in the **Enabled** column indicates that your Datagen Source has started generating sample data.



Configuring a datagen Source

Add a Destination

Each Cribl Stream Destination represents a data output. Options include Splunk, Kafka, Kinesis, InfluxDB, Snowflake, Databricks, TCP JSON, and others.

For this tutorial, we'll use Cribl Stream's built-in **DevNull** Destination. This simply discards events – not very exciting! But it simulates a real output, so it provides a configuration-free quick start for testing Cribl Stream setups. It's ideal for our purposes.

To verify that **DevNull** is enabled, let's walk through setting up a Destination, then setting it up as Cribl Stream's default output:

On Cribl.Cloud or any other distributed mode, first click the top nav's **Manage** tab to select the `default` (or another) Worker Group.

1. From Cribl Stream's top menu, select **Data** > **Destinations**.
2. From the **Data Destinations** page's tiles or left menu, select **DevNull**.

(You can use the search box to jump to the **DevNull** tile.)

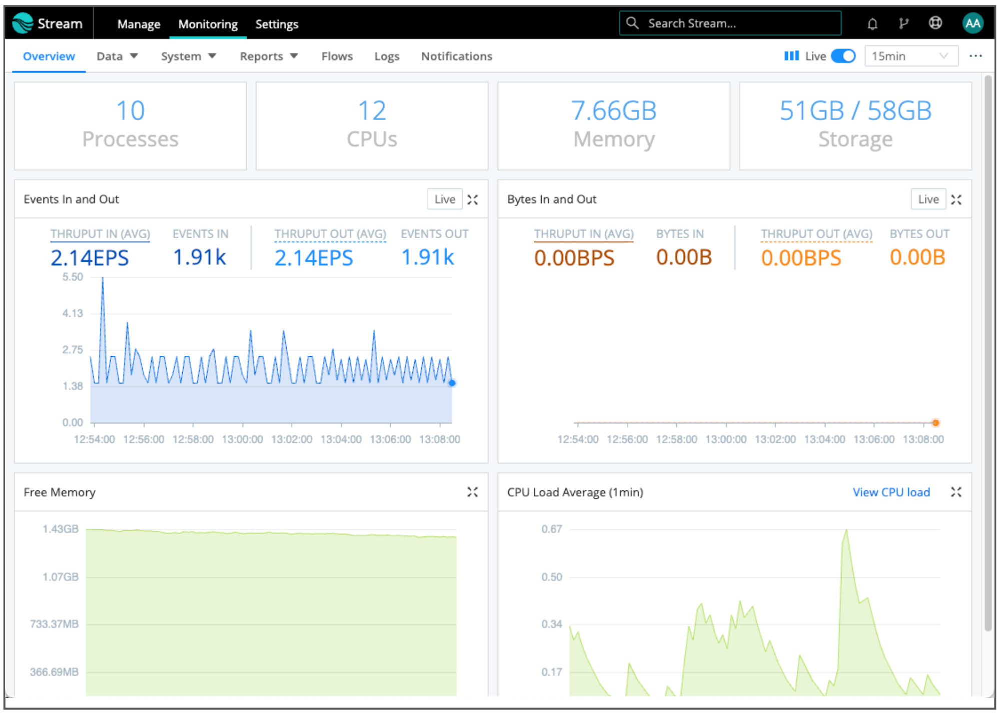
3. On the resulting **devnull** row, look for the **Live** indicator under **Status**. This confirms that the **DevNull** Destination is ready to accept events.
4. From the **Data Destinations** page's left nav, select the **Default** Destination at the top.
5. On the resulting **Manage Default Destination** page, verify that the **Default Output ID** drop-down points to the **devnull** Destination we just examined.

If you're on Cribl.Cloud or any other distributed mode, click **Commit & Deploy** at Cribl Stream's upper right (if available) before proceeding. Then, in the resulting dialog box, click **Commit & Deploy** to confirm. You'll see a **Commit successful** message.

We've now set up data flow on both sides. Is data flowing? Let's check.

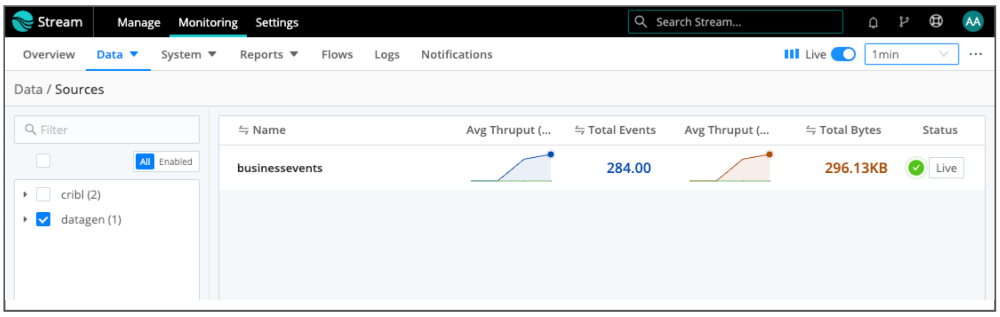
Monitor Data Throughput

Click the top nav's **Monitoring** tab. This opens a summary dashboard, where you should see a steady flow of data in and out of Cribl Stream. The left graph shows events in/out. The right graph shows bytes in/out.



Monitoring dashboard

Monitoring displays data from the preceding 24 hours. You can use the **Monitoring** submenu to open detailed displays of Cribl Stream components, collection jobs and tasks, and Cribl Stream's own internal logs. Click **Sources** on the lower submenu to switch to this view:

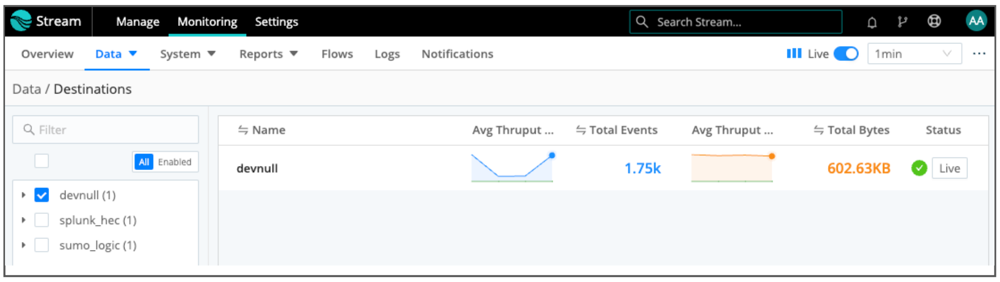


Name	Avg Thruput (...	Total Events	Avg Thruput (...	Total Bytes	Status
businessevents		284.00		296.13KB	Live

Monitoring Sources

This is a compact display of each Source's inbound events and bytes as a sparkline. You can click each Source's Expand button (highlighted at right) to zoom up detailed graphs.

Click **Destinations** on the lower submenu. This displays a similar sparklines view, where you can confirm data flow out to the **devnull** Destination:



Name	Avg Thruput ...	Total Events	Avg Thruput ...	Total Bytes	Status
devnull		1.75k		602.63KB	Live

Monitoring Destinations

With confidence that we've got data flowing, let's send it through a Cribl Stream Pipeline, where we can add Functions to refine the raw data.

Create a Pipeline

A Pipeline is a stack of Cribl Stream Functions that process data. Pipelines are central to refining your data, and also provide a central Cribl Stream workspace – so let's get one going.

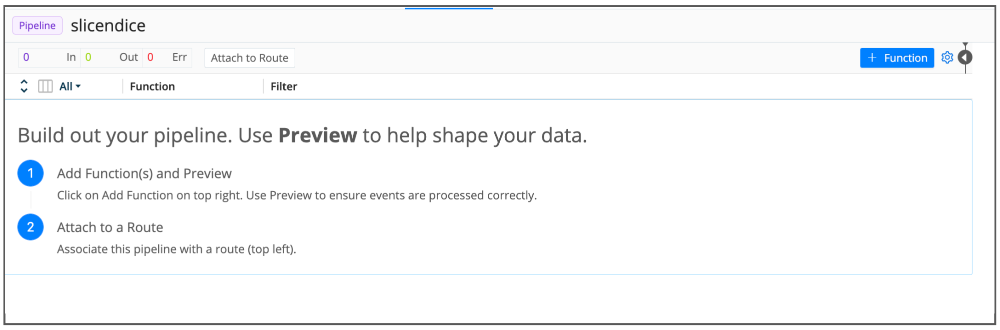
On Cribl.Cloud or any other distributed mode, first click the top nav's **Manage** tab to select the `default` (or another) Worker Group.

1. From the top menu, select **Processing** > **Pipelines**.
   You now have a two-pane view, with a **Pipelines** list on the left and **Sample Data** controls on the right. (We'll capture some sample data momentarily.)
2. At the **Pipelines** pane's upper right, click **Add Pipeline**, then select **Create Pipeline**.
3. In the new Pipeline's **ID** field, enter a unique identifier. (For this tutorial, you might use `slicendice`.)
4. Optionally, enter a **Description ** of this Pipeline's purpose.
5. Click **Save**.

Now scroll through the right **Preview** pane. Depending on your data sample, you should now see multiple events struck out and faded – indicating that Cribl Stream will drop them before forwarding the data.

If you're on Cribl.Cloud or any other distributed mode, click **Commit & Deploy** at Cribl Stream's upper right before proceeding. Then, in the resulting dialog box, click **Commit & Deploy** to confirm. You'll see a **Commit successful** message.

Your empty Pipeline now prompts you to preview data, add Functions, and attach a Route. So let's capture some data to preview.



Build out your pipeline. Use **Preview** to help shape your data.

1. Add Function(s) and Preview
   Click on Add Function on top right. Use Preview to ensure events are processed correctly.
2. Attach to a Route
   Associate this pipeline with a route (top left).

Pipeline prompt to add Functions

Capture and Filter Sample Data

The right **Sample Data** pane provides multiple tools for grabbing data from multiple places (inbound streams, copy/paste, and uploaded files); for previewing and testing data transformations as you build them; and for saving and reloading sample files.

Since we've already got live (simulated) data flowing in from the datagen Source we built, let's grab some of that data.

Capture New Data

1. In the right pane, click **Capture Data**.
2. Click **Capture**, then accept the drop-down's defaults – click **Start**.
3. When the modal finishes populating with events, click **Save as Sample File**.
4. In the **SAMPLE FILE SETTINGS** fly-out, change the generated **File Name** to a name you'll recognize, like `be_raw.log`.
5. Click **Save**. This saves to the **File Name** you entered above, and closes the modal. You're now previewing the captured events in the right pane. (Note that this pane's **Simple Preview** tab now has focus.)
6. Click the **Show more** link to expand one or more events.

By skimming the key-value pairs within the data's `_raw` fields, you'll notice the scenario underlying this preview data (provided by the `businessevents.log` datagen): these are business logs from a mobile-phone provider.

To set up our next step, find at least one `marketState` K=V pair. Having captured and examined this raw data, let's use this K=V pair to crack open Cribl Stream's most basic data-transformation tool, Filtering.

Filter Data and Manage Sample Files

1. Click the right pane's **Sample Data** tab.
2. Again click **Capture New**.
3. In the **Capture Sample Data** modal, replace the **Filter Expression** field's default `true` value with this simple regex: `_raw.match(/marketState=TX/)`
   We're going to Texas! If you type this in, rather than pasting it, notice how Cribl Stream provides typeahead assist to complete a well-formed JavaScript expression.
   You can also click the Expand button at the **Filter Expression** field's right edge to open a modal to validate your expression. The adjacent drop-down enables you to restore previously used expressions.

Expand button and history drop-down

4. Click **Capture**, then **Start**.
   Using the **Capture** drop-down's default limits of 10 seconds and 10 events, you'll notice that with this filter applied, it takes much longer for Cribl Stream to capture 10 matching events.
5. Click **Cancel** to discard this filtered data and close the modal.
6. On the right pane's **Sample Data** tab, click **Simple** beside `be_raw.log`.

This restores our preview of our original, unfiltered capture. We're ready to transform this sample data in more interesting ways, by building out our Pipeline's Functions.

Refine Data with Functions

Functions are pieces of JavaScript code that Cribl Stream invokes on each event that passes through them. By default, this means all events – each Function has a **Filter** field whose value defaults to `true`. As we just saw with data capture, you can replace this value with an expression that scopes the Function down to particular matching events.

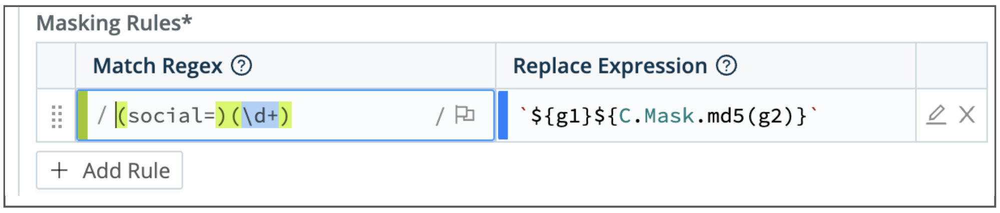
In this Pipeline, we'll use some of Cribl Stream's core Functions to:

- Redact (mask) sensitive data
- Extract (parse) the `_raw` field's key-value pairs as separate fields.
- Add a new field.
- Delete the original `_raw` field, now that we've extracted its contents.
- Rename a field for better legibility.

Mask: Redact Sensitive Data

In the right **Preview** pane, notice each that event includes a **social** key, whose value is a (fictitious) raw Social Security number. Before this data goes any further through our Pipeline, let's use Cribl Stream's `Mask` Function to swap in an md5 hash of each SSN.

1. In the left **Pipelines** pane, click **Add Function**.
2. Search for `Mask`, then click it.
3. In the new Function's **Masking Rules**, click the into **Match Regex** field.
   Now we want to build the **Match Regex/Replace Expression** row shown just below.
4. Enter or paste this regex, which simply looks for the string `social=`, followed by any digits:
   `(social=)(\d+)`
5. In **Replace Expression**, paste the following hash function. The backticks are literal:
   ``` `${g1}${C.Mask.md5(g2)}` ```
6. Note that **Apply to Fields** defaults to `_raw`. This is what we want to target, so we'll accept this default.
7. Click **Save**.


Masking Rules*

	Match Regex	Replace Expression
	/ (social=)(\d+) /	`${g1}${C.Mask.md5(g2)}`

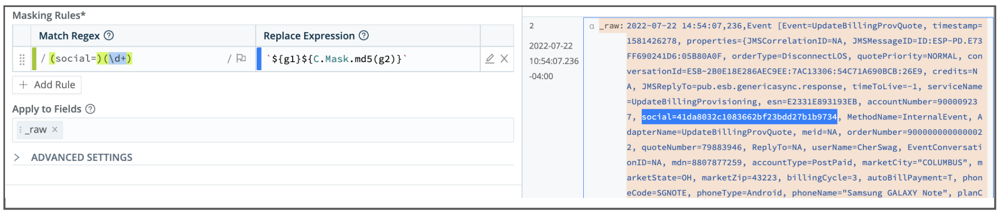
Add Rule

Entering a Masking Rule to hash Social Security numbers

You'll immediately notice some obvious changes:

- The **Preview** pane has switched from its **IN** to its **OUT** tab, to show you the outbound effect of the Pipeline you just saved.
- Each event's `_raw` field has changed color, to indicate that it's undergone some redactions.

Now locate at least one event's **Show more** link, and click to expand it. You can verify that the `social` values have now been hashed.


Masking Rules*

	Match Regex	Replace Expression
	/ (social=)(\d+) /	`${g1}${C.Mask.md5(g2)}`

Apply to Fields: _raw

Add Rule
ADVANCED SETTINGS

```
2
2022-07-22
10:54:07.236
-04:00
```

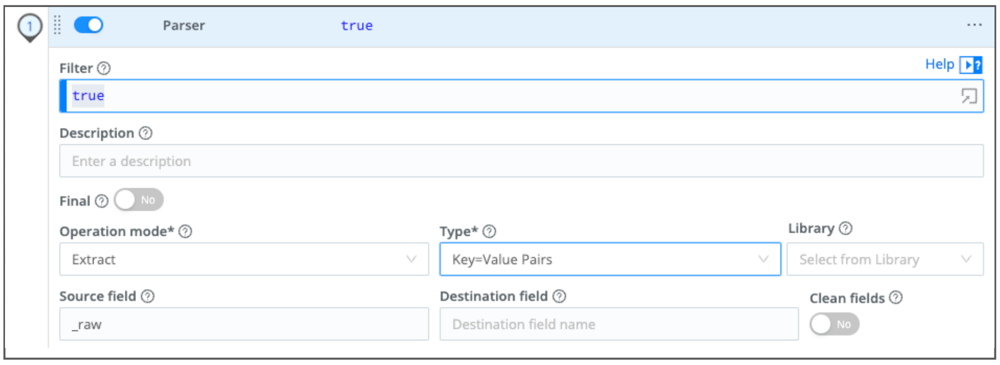
```
_raw: 2022-07-22 14:54:07,236,Event [Event=UpdateBillingProvQuote, timestamp=
      1581426278, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.E73
      FF690241D6:05B80A0F, orderType=DisconnectLOS, quotePriority=NORMAL, con
      versationId=ESB~2B0E18E286AEC9EE:7AC13306:54C71A690BCB:26E9, credits=N
      A, JMSReplyTo=pub.esb.genericasync.response, timeToLive=-1, serviceName
      =UpdateBillingProvisioning, esn=E2331E893193EB, accountNumber=90000923
      7, social=41da8032c1083662bf23bdd27b1b9734, MethodName=InternalEvent, A
      dapterName=UpdateBillingProvQuote, meid=NA, orderNumber=900000000000002
      2, quoteNumber=79883946, ReplyTo=NA, userName=CherSwag, EventConversati
      onID=NA, mdn=8807877259, accountType=PostPaid, marketCity="COLUMBUS", m
      arketState=OH, marketZip=43223, billingCycle=3, autoBillPayment=T, phon
      eCode=SGNOTE, phoneType=Android, phoneName="Samsung GALAXY Note", planC
```

Mask Function and hashed result
```

Parser: Extract Events

Having redacted sensitive data, we'll next use a Parser function to lift up all the `_raw` field's key-value pairs as fields:

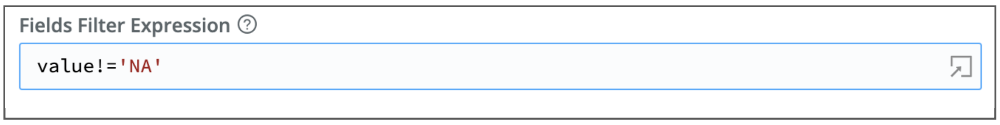
1. In the left **Pipelines** pane, click **Add Function**.
2. Search for `Parser`, then click it.
3. Leave the **Operation Mode** set to its `Extract` default.
4. Set the **Type** to `Key=Value Pairs`.
5. Leave the **Source Field** set to its `_raw` default.
6. Click **Save**.


Parser true
Filter: true
Description: Enter a description
Final: No
Operation mode*: Extract
Type*: Key=Value Pairs
Library: Select from Library
Source field: _raw
Destination field: Destination field name
Clean fields: No

Parser configured to extract K=V pairs from `_raw`

You should see the **Preview** pane instantly light up with a lot more fields, parsed from `_raw`. You now have rich structured data, but not all of this data is particularly interesting: Note how many fields have `NA` ("Not Applicable") values. We can enhance the **Parser** Function to ignore fields with `NA` values.

1. In the Function's **Fields Filter Expression** field (near the bottom), enter this negation expression: `value!='NA'`.
   Note the single-quoted value. If you type (rather than paste) this expression, watch how typeahead matches the first quote you type.
2. Click **Save**, and watch the **Preview** pane.


Fields Filter Expression
```
value!='NA'
```

Filtering the Parser Function to ignore fields with 'NA' values

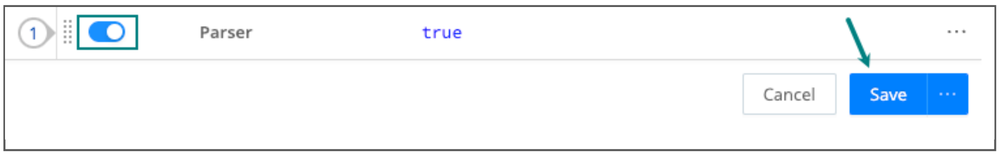
Several fields should disappear – such as `credits`, `EventConversationID`, and `ReplyTo`. The remaining fields should display meaningful values. Congratulations! Your log data is already starting to look better-organized and less bloated.

Missed It?

If you didn't see the fields change, toggle the Parser Function `Off`, click **Save** below, and watch the **Preview** pane change. Using these toggles, you can preserve structure as you test and troubleshoot each Function's effect.

Note that each Function also has a **Final** toggle, defaulting to `Off`. Enabling **Final** anywhere in the Functions stack will prevent data from flowing to any Functions lower in the UI.

Be sure to toggle the Function back `On`, and click **Save** again, before you proceed!



Toggling a Function off and on

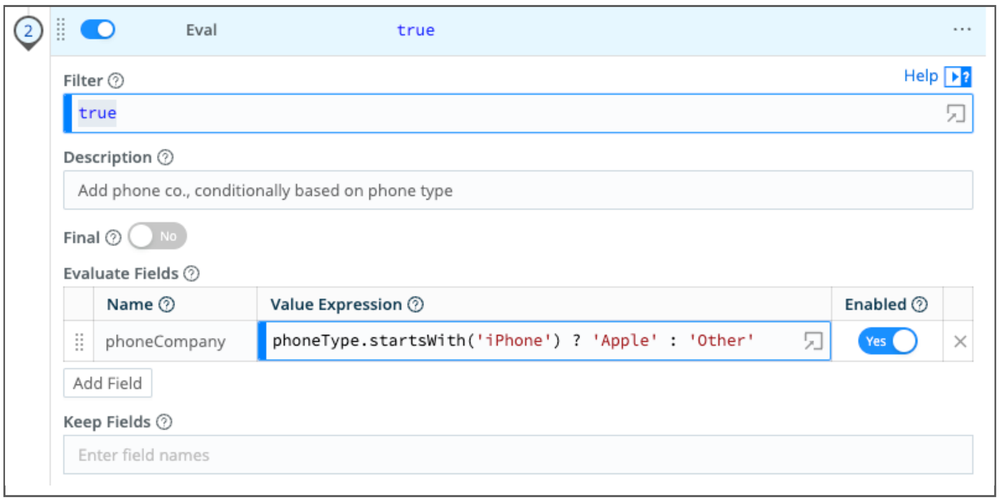
Next, let's add an extra field, and conditionally infer its value from existing values. We'll also remove the `_raw` field, now that it's redundant. To add and remove fields, the **Eval** Function is our pal.

Eval: Add and Remove Fields

Let's assume we want to enrich our data by identifying the manufacturer of a certain popular phone handset. We can infer this from the existing `phoneType` field that we've lifted up for each event.

Add Field (Enrich)

1. In the left **Pipelines** pane, click **Add Function**.
2. Search for `Eval`, then click it.
3. Click **Add Field** to open the **Evaluate Fields** table.
   Here you add new fields to events, defining each field as a key-value pair. If we needed more key-value pairs, we could click `+ Add Field` for more rows.
4. In the table's first row, click into the **Name** field and enter: `phoneCompany`.
5. In the adjacent **Value Expression** field, enter this JS ternary expression that tests `phoneType`'s value: `phoneType.startsWith('iPhone') ? 'Apple' : 'Other'` (Note the `?` and `:` operators, and the single-quoted values.)
6. Click **Save**. Examine some events in the **Preview** pane, and each should now contain a `phoneCompany` field that matches its `phoneType`.



Eval true

Filter: `true`

Description: Add phone co., conditionally based on phone type

Final: No

Evaluate Fields

Name	Value Expression	Enabled
phoneCompany	`phoneType.startsWith('iPhone') ? 'Apple' : 'Other'`	Yes

Keep Fields: Enter field names

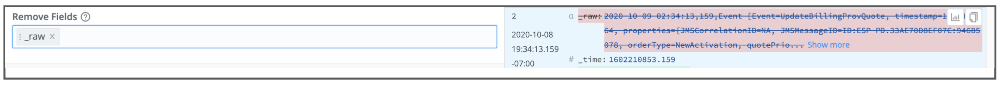
Adding a field to enrich data

Remove Field (Shrink Data)

Now that we've parsed out all of the `_raw` field's data – it can go. Deleting a (large) redundant field will give us cleaner events, and reduced load on downstream resources.

1. Still in the **Eval** Function, click into **Remove Fields**.
2. Type: `_raw` and press **Tab** or **Enter**.
3. Click **Save**.

The **Preview** pane's diff view should now show each event's `_raw` field stripped out.



Remove Fields: _raw

```
2           α ~~_raw: 2020-10-09 02:34:13,159,Event [Event=UpdateBillingProvQuote, timestamp=1
2020-10-08      64, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.33AE70D8EF07C:946B5
19:34:13.159    078, orderType=NewActivation, quotePrio...~~
-07:00      # _time: 1602210853.159
```

Removing a field to streamline data

Our log data has now been cleansed, structured, enriched, and slimmed-down. Let's next look at how to make it more legible, by giving fields simpler names.

Rename: Refine Field Names

1. In the left **Pipelines** pane, click `Add Function`.
   This rhythm should now be familiar to you.
2. Search for `Rename`, then click it.
3. Click **Add fields** to open the **Rename fields** table.
4. Click into the new Function's **Rename fields** table.
   This has the same structure you saw above in Eval: Each row defines a key-value pair.
5. In **Current name**, enter the longhaired existing field name: `conversationId`.

6. In **New name**, enter the simplified field name: `ID`.
7. Watch any event's `conversationId` field in the **Preview** pane as you click **Save** at left. This field should change to `ID` in all events.

Drop: Remove Unneeded Events

We've already refined our data substantially. To further slim it down, a Pipeline can entirely remove events that aren't of interest for a particular downstream service.

As the "Pipeline" name implies, your Cribl Stream installation can have multiple Pipelines, each configured to send out a data stream tailored to a particular Destination. This helps you get the right data in the right places most efficiently.

Here, let's drop all events for customers who use prepaid monthly phone service (i.e., **not** postpaid):

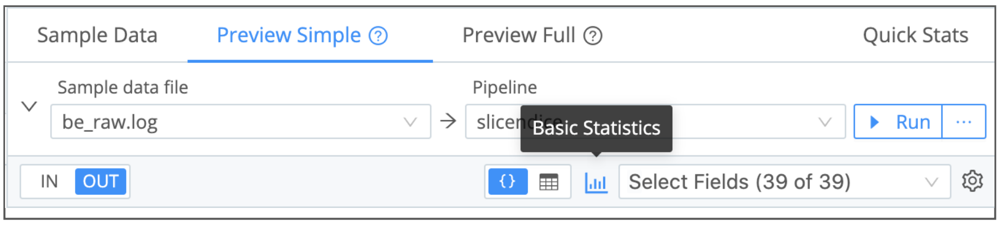
1. In the left **Pipelines** pane, click **Add Function**.
2. Search for `Drop`, then click it.
3. Click into the new Function's **Filter** field.
4. Replace the default `true` value with this JS negation expression: `accountType!='PostPaid'`
5. Click **Save**.

Now scroll through the right **Preview** pane. Depending on your data sample, you should now see multiple events struck out and faded – indicating that Cribl Stream will drop them before forwarding the data.

A Second Look at Our Data

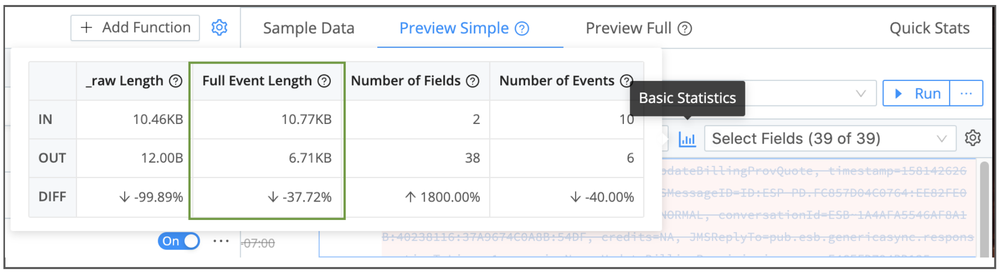
Torture the data enough, and it will confess. By what factor have our transformations refined our data's volume? Let's check.

In the right **Preview** pane, click the **Basic Statistics** button:



Displaying Basic Statistics

Even without the removal of the `_raw` field (back in Eval) and the dropped events, you should see a substantial % reduction in the **Full Event Length**.



	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	10.46KB	10.77KB	2	10
OUT	12.00B	6.71KB	38	6
DIFF	↓ -99.89%	↓ -37.72%	↑ 1800.00%	↓ -40.00%

Data reduction quantified

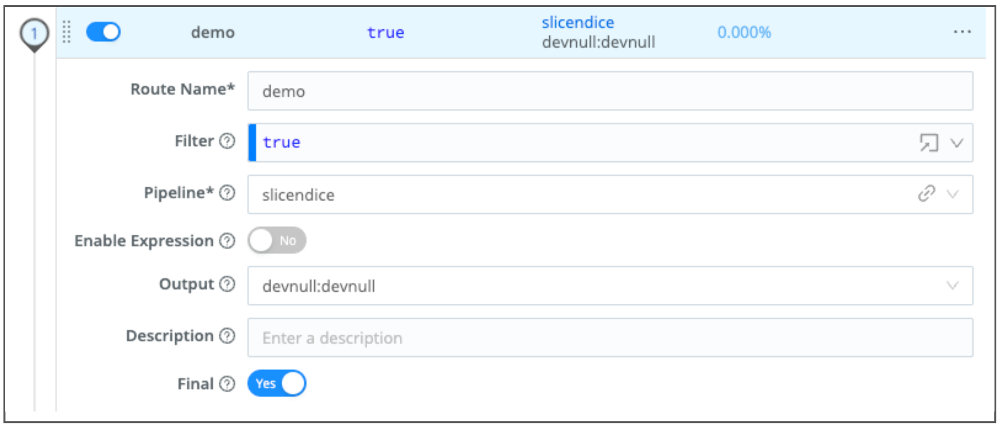
Woo hoo! Before we wrap up our configuration: If you're curious about individual Functions' independent contribution to the data reduction shown here, you can test it now. Use the toggle `Off` > **Save** > **Basic Statistics** sequence to check various changes.

Add and Attach a Route

We've now built a complete, functional Pipeline. But so far, we've tested its effects only on the static data sample we captured earlier. To get dynamic data flowing through a Pipeline, we need to filter that data in, by defining a Cribl Stream Route.

1. At the **Pipelines** page's top left, click **Attach to Route**.
   This displays the **Routes** page. It's structured very similarly to the **Pipelines** page, so the rhythm here should feel familiar.
2. Click `Add Route`.
3. Enter a unique, meaningful **Route Name**, like `demo`.
4. Leave the **Filter** field set to its `true` default, allowing it to deliver all events.
   Because a Route delivers events to a Pipeline, it offers a first stage of filtering. In production, you'd typically configure each Route to filter events by appropriate `source`, `sourcetype`, `index`, `host`, `_time`, or other characteristics. The **Filter** field accepts JavaScript expressions, including AND (`&&`) and OR (`||`) operators.
5. Set the **Pipeline** drop-down to our configured `slicendice` Pipeline.
6. Leave the **Enable Expression** field set to its `No` default. Toggling this field to `Yes` changes the **Output** field to an **Output Expression** field where you can enter a JavaScript expression for your **Destination** name.
7. Set the **Output** drop-down to either `devnull` or `default`.
   This doesn't matter, because we've set `default` as a pointer to `devnull`. In production, you'd set this carefully.
8. You can leave the **Description** empty, and leave **Final** set to `Yes`.
9. Grab the new Route by its left handle, and drag it above the `default` Route, so that our new Route will process events first. You should see something like the screenshot below.
10. Click **Save** to save the new Route to the Routing table.

If you're on Cribl.Cloud or any other distributed mode, click **Commit & Deploy** at Cribl Stream's upper right before proceeding. Then, in the resulting dialog box, click **Commit & Deploy** to confirm. You'll see a **Commit successful** message.

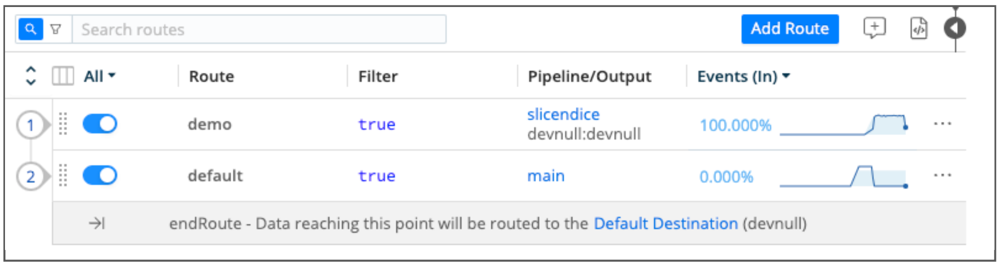


Route	Filter	Pipeline/Output	Events (In)
1 demo	true	slicendice devnull:devnull	0.000%

Route Name*	demo
Filter	true
Pipeline*	slicendice
Enable Expression	No
Output	devnull:devnull
Description	Enter a description
Final	Yes

Configuring and adding a Route

The sparklines should immediately confirm that data is flowing through your new Route:

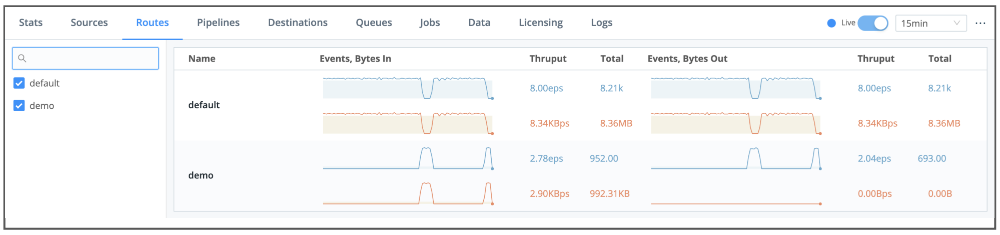


	Route	Filter	Pipeline/Output	Events (In)
1	demo	true	slicendice devnull:devnull	100.000%
2	default	true	main	0.000%

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Live Routes

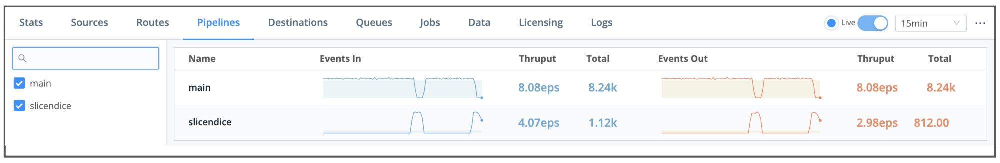
To confirm data flow through the whole system we've built, select **Monitoring** > **Data** > **Routes** and examine `demo`.



Name	Events, Bytes In	Thruput	Total	Events, Bytes Out	Thruput	Total
default		8.00eps	8.21k		8.00eps	8.21k
		8.34KBps	8.36MB		8.34KBps	8.36MB
demo		2.78eps	952.00		2.04eps	693.00
		2.90KBps	992.31KB		0.00Bps	0.00B

Monitoring data flow through Routes

Also select **Monitoring** > **Data** > **Pipelines** and examine `slicendice`.



Name	Events In	Thruput	Total	Events Out	Thruput	Total
main		8.08eps	8.24k		8.08eps	8.24k
slicendice		4.07eps	1.12k		2.98eps	812.00

What Have We Done?

Look at you! Give yourself a pat on the back! In this short, scenic tour – with no hit to your cloud-services charges – you've built a simple but complete Cribl Stream system, exercising all of its basic components:

- Downloaded, installed, and run Cribl Stream.
- Configured a Source to hook up an input.
- Configured a Destination to feed an output.
- Monitored data throughput, and checked it twice.
- Built a Pipeline.
- Configured Cribl Stream Functions to redact, parse, enrich, trim, rename, and drop event data.
- Added and attached a Route to get data flowing through our Pipeline.

Next Steps

Interested in guided walk-throughs of more-advanced Cribl Stream features? We suggest that you next check out these further resources.

- Cribl Stream Sandboxes: Work through general and specific scenarios in a free, hosted environment, with terminal access and real data inputs and outputs.
- Distributed Quick Start: Building on this tutorial that you've just completed, launch and configure a Cribl Stream distributed deployment. You'll work with a small but realistic model of a fully scaleable production deployment.
- Use Cases documentation: Bring your own services to build solutions to specific challenges.
- Cribl Concept: Pipelines – Video showing how to build and use Pipelines at multiple Cribl Stream stages.
- Cribl Concept: Routing – Video about using Routes to send different data through different paths.

Cleaning Up

Oh yeah, you've still got the Cribl Stream server running, with its `businessevent.log` datagen still firing events. If you'd like to shut these down for now, in reverse order:

1. Go to **Data** > **Sources** > **Datagen**.
2. Toggle `businessevent` to `Off`, and click **Save**. (Refer back to the screenshot above.)

3. In your terminal’s `$CRIBL_HOME/bin` directory, shut down the server with: `./cribl stop`

That’s it! Enjoy using Cribl Stream.

1.4. Distributed Quick Start

This tutorial builds on our Getting Started Guide by walking you through a distributed deployment – Cribl Stream's typical deployment type for production.

> For concepts and deeper details underlying the techniques presented here, see Distributed Deployment.

Requirements

To exercise these distributed features, you'll need the following prerequisites.

Licenses and Instances

This tutorial is tiered to accommodate different Cribl Stream license types:

- With a Cribl Stream Free, One, or Standard license, you'll install three Cribl Stream instances on one or multiple physical or virtual machines – one Leader Node, and two Worker Nodes.
- To do the **optional** section on adding and managing multiple Worker Groups, you'll need an Enterprise or Sales Trial license/plan.

Basic Setup

Basic building blocks are identical to the Getting Started Guide. Please refer to the following sections of that tutorial for details, as needed:

- (System) Requirements
- Download and Install Cribl Stream
- Run Cribl Stream

To set up the multiple instances you'll need, choose among the options below in Three Instances, Pick Any Medium. But first, let's lay out the division of labor in a single Worker Group.

Distributed Deployment (Identical Workers)

A distributed deployment enables Cribl Stream to scale out to handle higher data volumes, load-balancing with failover, and parallel data processing based on conditional mapping and routing.

A single Leader Node manages multiple Worker Nodes. The Leader distributes and updates configuration on the Workers, handles version control, and monitors the Workers' health and activity metrics. The Workers do all the data processing.

Here, we'll show how this works by configuring a Leader and two Workers. This configuration is compact, and can be demonstrated without an Enterprise (or other paid) license/plan. But you can extrapolate the same technique to setting up enterprise-scale deployments of hundreds of Workers, to handle petabytes of data.

Cribl Stream Free, One, and Standard licenses support only a single Worker Group (named `default`), so in this example, all Workers share identical configuration. With an Enterprise license/plan, you can organize Workers into multiple Groups, with varying configurations to handle scenarios like on-premises versus cloud tech stacks, or data centers in different geographic locations. For details, see Worker Groups – What Are They and Why You Should Care.

Three Instances, Pick Any Medium

You'll need to deploy three Cribl Stream instances – with SSH access – on one or more physical machines, virtual machines, or containers. If you haven't already provisioned this infrastructure, you have several alternatives, listed in the following subsections:

- Docker Containers
- Curl
- Amazon Lightsail
- AWS/EC2 (CloudFormation Optional)
- Kubernetes/Helm

Pick whichever approach will make it easiest for you to get the infrastructure launched – based on familiarity, preference, or availability.

We Don't Need No Stinkin' Permissions Errors!

Cribl's Docker containers come with Cribl Stream preinstalled. If you select any other option, be sure to both install and run Cribl Stream as the same Linux user. (For details on creating a new user – addressing both systemd and initd distro's – see Enabling Start on Boot.)

Docker Containers

You can use the Docker Compose (`docker-compose.yml`) file below to easily stand up a Cribl Stream distributed deployment of a Leader and multiple Workers on one machine.

Before you use the Docker Compose file

1. Open the `docker-compose.yml` file in a text editor.
2. Replace both instances of `INSERT_TOKEN` with a unique, secure token value.
3. Save the updated `docker-compose.yml` file.

docker-compose.yml

```
version: '3.8'
services:
  master:
    image: cribl/cribl:latest
    environment:
      - CRIBL_DIST_MODE=master
      - CRIBL_DIST_MASTER_URL=tcp://INSERT_TOKEN@0.0.0.0:4200
      - CRIBL_VOLUME_DIR=/opt/cribl/config-volume
    ports:
      - "19000:9000"
    volumes:
      - "~/cribl-config:/opt/cribl/config-volume"
  workers:
    image: cribl/cribl:latest
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://INSERT_TOKEN@master:4200
    ports:
      - 9000
```

This uses a local directory, `~/cribl-config`, as the configuration store for Cribl Stream. You must create this directory (it can be empty) before you run the `docker-compose` command.

If you prefer to use ephemeral storage, you can delete line 8 (the `CRIBL_VOLUME_DIR` definition) and lines 11–12 (the `volumes` configuration) before running the `docker-compose` command. But this will make it hard to stop and restart the same infrastructure, if you want to do the tutorial in chunks.

To deploy a Leader Node, plus (e.g.) two Workers already configured and wired up to the Leader, use this command:

```
docker-compose up -d --scale workers=2
```

To deploy a different number of Workers, just change the `workers=2` value. By default, the above command pulls the freshest stable image (tagged `cribl/cribl:latest`) from Cribl's Docker Hub.
It defaults to the following URLs and ports:

- Leader URL: `http://localhost:19000`
- Worker URLs: `http://localhost:<automatically-assigned-host-ports>`

If you're running the container itself on a virtual machine, replace `localhost` with the VM's IP address. The automatic assignment of available host-OS ports to the Workers prevents port collisions. **Within** the Docker container, these ports will forward over TCP to port 9000. To see the ports assigned on the OS, enter:

```
docker ps
```

You should see results like these:

```
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATU
a3de9ea8f46f   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
40aa687baefc   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
df362a65f7d1   cribl/cribl:latest   "/sbin/entrypoint.sh…"   13 seconds ago   Up 11
```

The **PORTS** column shows the host-OS ports on the left, forwarding to the container-internal ports on the right. You can use the `docker_workers_N` ports if you want to log directly into Workers. In the above example:

- Worker1 URL: `http://localhost:63411`
- Worker2 URL: `http://localhost:63410`

If your Leader is crashing with two Workers, make sure you are allocating enough memory to Docker.

Once your three instances are running, proceed to Configure Leader Instance.

Curl

Use our Download page's `curl` command to directly install Cribl Stream onto your chosen infrastructure.

For x64 processors, use: `curl -Lso - $(curl https://cdn.cribl.io/dl/latest-x64) | tar zxv`

For ARM64 processors: `curl -Lso - $(curl https://cdn.cribl.io/dl/latest-arm64) | tar zxv`

Once you've configured the first Cribl Stream instance as a Leader, you can bootstrap Workers from the Leader. Or you can create Workers (with tags) from the Leader, using a `curl` command of this form:

```
curl 'http://<leader-ip-or-hostname>:9000/init/install-worker.sh?token=<token>&tag=<tag1>&tag=<tag2>'
```

E.g.: `curl 'http://localhost:9000/init/install-worker.sh?token=<token>&tag=dev&tag=test''`

Once your three instances are running, proceed to Configure Leader Instance.

Amazon Lightsail

Amazon Lightsail provides a quick, simple way to deploy Cribl Stream to AWS instances. Amazon's Get Started with Linux/Unix-based Instances in Amazon Lightsail tutorial walks you through setting up your instances and connecting via SSH.

- The free (first month) tier is all you need for this tutorial.
- As the "platform," Cribl recommends selecting **Amazon Linux 2 (default CentOS)**.
- Lightsail doesn't support IAM roles assigned to instances, or advanced load balancing, but it's adequate for this tutorial, which is not a production deployment.

Once your three instances are running, proceed to Configure Leader Instance.

AWS/EC2 (CloudFormation Optional)

You can deploy Cribl Stream's AWS EC2 instances using Cribl's CloudFormation template, see our AWS/EC2 Quick Start Guide on GitHub. Follow the **Cribl Stream Distributed** instructions. (You'll be responsible for the costs of your AWS infrastructure.)

If you prefer to deploy your own EC2 instances, the free tier is fine for this tutorial. Cribl recommends selecting **Amazon Linux 2 (default CentOS)** AMIs. Relevant instructions are linked below.

1. See any of these AWS Docs for EC2 deployment:

    - Tutorial: Getting Started with Amazon EC2 Linux Instances
    - Launching an Instance Using the Old Launch Instance Wizard
    - How Can I Create and Connect to an Amazon Linux EC2 Instance?
2. See these AWS instructions for SSH access:
    - Connect to Your Linux Instance Using an SSH Client

Once your three instances are running, proceed to Configure Leader Instance.

Kubernetes/Helm

Use Cribl's Helm charts to deploy Kubernetes pods, then proceed to the next section:

- Kubernetes Leader Deployment
- Kubernetes Worker Deployment

Configure Leader Instance

Once you've set up your Leader and Worker instances via your chosen approach above, you're ready to configure these instances and their communication.

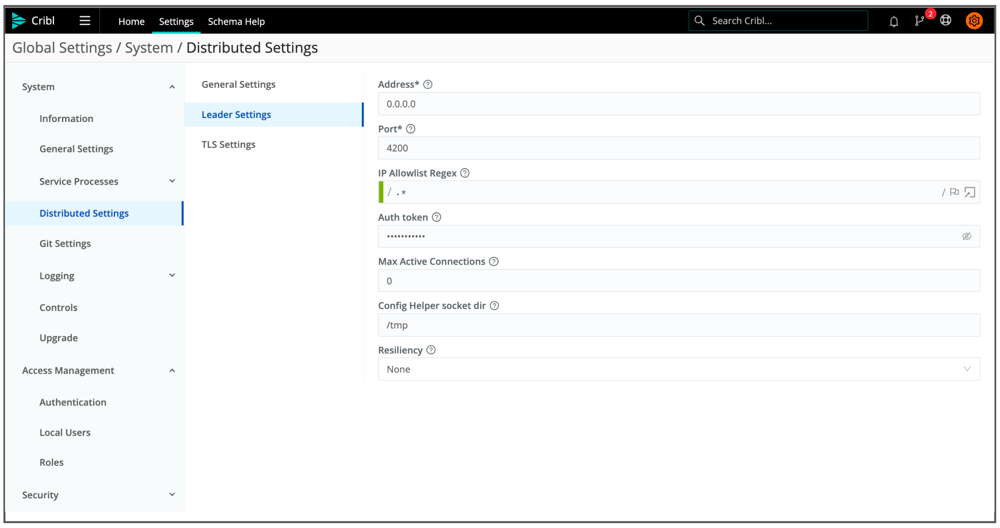
If you used an installation option like Docker Containers above, it has already preconfigured all three instances for you. This section and the next Configure Workers section will show you how to verify, and/or modify, these preset configurations.

If you want to jump ahead, your next required configuration step is Add a Source, further down.

Configure the first instance in Leader mode, and open port 4200.

1. Log into the Leader. Depending on the deployment method you chose, this will be at `http://<localhost-or-IP-address>:9000` or `http://<localhost-or-IP-address>:19000`. Use the default `admin/admin` credentials.
2. Complete the registration form.
3. From the UI's **Settings** > **Global Settings** > **System** > **Distributed Settings** > **General Settings**, select **Mode**: `Leader`.
4. Click the **Leader Settings** left tab, and make sure the **Port** is set to `4200`. (This port must be open for Workers to communicate with the Leader.)
5. Click **Save** to restart the Cribl server in Leader mode.

Keep the Leader's tab open, so that we can verify connectivity with the Workers after configuring them (in the next section).



Address* 0.0.0.0
Port* 4200
IP Allowlist Regex / .* /
Auth token •••••••••••
Max Active Connections 0
Config Helper socket dir /tmp
Resiliency None

Distributed > Leader Settings – the whole enchilada

6. Optional but recommended: From Cribl Stream's top nav, click **Manage** > **Groups**. Then enable the **UI access** toggle for your Group(s). This way, you will be able to click through from the Leader's **Manage** > **Workers** tab to view and manage each Worker's UI, distinguished by a purple border. (This option handles authentication for you, so you don't need to manage or enter the Workers' credentials.)

Configure Workers

Next, configure the two other instances as Workers that report to the Leader to receive processing instructions. We'll configure one instance through the UI, and (optionally) bootstrap the other from the Leader.

Configure Worker via UI

1. Log into the first Worker instance. Depending on the deployment method you chose, this will be at `http://<localhost-or-IP-address>:9000` or at: `http://<localhost-or-IP-address>:<automatically-assigned-host-port>`. Use the default `admin`/`admin` credentials.
2. Complete the registration form, if displayed.
3. From the UI's **Settings** > **Global Settings** > **Distributed Settings** > **Distributed Management** > **General Settings**, select **Mode**: `Stream: Managed Worker (managed by Leader)`.
4. Leave other settings on this tab unchanged, including **Default Group**: `default`.
   (This is the literal name of the single Worker Group available with free licenses.)
5. On the **Leader Settings** tab, make sure the **Address** matches the domain of the Leader you previously configured.

6. On the same **Leader Settings** tab, display the **Auth token** value. Optionally, you can change this from the default.
7. Leave other settings unchanged, and click **Save** to restart this instance as a managed Worker.

Bootstrap Worker from Leader

You can (if you choose) configure the second Worker instance using exactly the same procedure you used just above. But here, we'll offer you a simplified version of Cribl Stream's Bootstrapping Workers from Leader procedure for downloading the config from the Leader to the second Worker:

First, switch to the terminal/console of the instance you've reserved for this second Worker.

Next, if you didn't change the default **Auth token** value when you previously configured the Leader, run this command as root user:

```
curl http://<leader-hostname-or-IP>:9000/init/install-worker.sh | sh -
```

If you cannot run as root, insert `sudo` as you pipe to the shell:

```
curl http://<leader-hostname-or-IP>:9000/init/install-worker.sh | sudo sh -
```

To instead pipe to a bash shell:

```
curl http://<leader-hostname-or-IP>:9000/init/install-worker.sh | [sudo] bash -
```

If you substituted a custom **Auth token** value on the Leader, enter:

```
curl http://<leader-hostname-or-IP>:9000/init/install-worker.sh?token=<your-custom-
```

Or, for bash:

```
curl http://<leader-hostname-or-IP>:9000/init/install-worker.sh?token=<your-custom-
```

The bootstrap script will install Cribl Stream into `/opt/cribl` on the target instance.

Verify Workers' Communication with the Leader

With both Workers configured, next make sure they're visible to the Leader.

1. Switch back to Cribl Stream's UI on the Leader instance you previously configured.
2. From Cribl Stream's top nav, click **Manage**.
3. Below that, click the **Workers** tab.
4. On the resulting **Manage Workers** page, you should now see both Workers, mapped to the `default` **Group**.

If one or both Workers are missing, repeat the preceding Configure Worker via UI and/or Bootstrap Worker from Leader procedures until the Workers show up.

Otherwise, if both Workers are present, we can now configure a few more resources to get data flowing through this distributed setup.

If you're interested in details about the communication among Cribl Stream instances, see How Do Workers and Leader Work Together.

Once you've configured a Worker to point to the Leader Node, the Leader will assign the Worker a new, random admin password. This secures each Worker from unintended access. If you need to reconfigure the Worker later, you can either:

- Enable the Leader's **UI access** option, as recommended above; or
- Reset the password on the Worker Group – see Set Worker Passwords.

Add a Source

To minimize dependencies, this section walks you through enabling a Cribl Stream built-in Datagen Source to get some (fake) events flowing into your Workers. (This is the same approach used in our single-instance Getting Started Guide tutorial, but using a different datagen.)

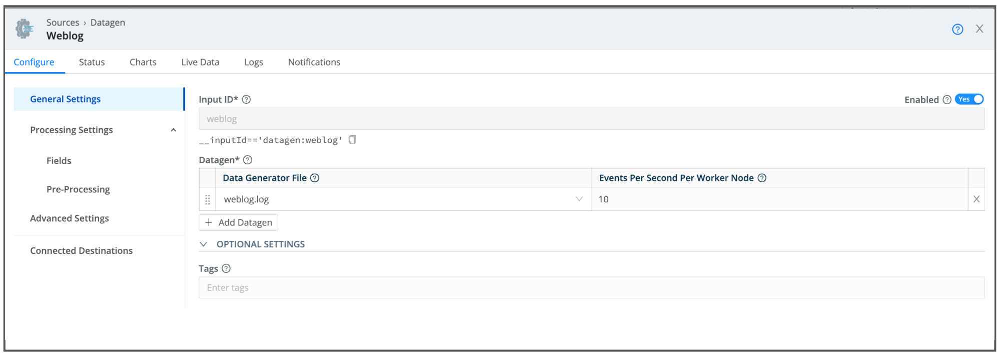
If you prefer to configure Cribl Stream to receive events from a real data input that you've already set up, see our Sources topic for a link to the appropriate instructions.

1. In the Leader UI's top nav, click **Manage** > **Groups**.
2. From the resulting Groups page, click the Group name.
   (Working with a free license, we're implicitly configuring this Source on the `default` Group.)
3. From the top nav, select **Manage** > **Data** > **Sources**.

4. From the **Data Sources** page's tiles or left menu, select **Datagen**.
   (You can use the search box to jump to the **Datagen** tile.)
5. Click **Add Source** to open a **New Source** modal.
6. In the **Input ID** field, name this Source `weblog` (or any unique name).
7. In the **Data Generator File** drop-down, select `weblog.log`.
   This generates...simulated log events for a website.
8. Keep the **Events per Second per Worker Node** at the default `10` EPS for now.
9. Click **Save**.

In the **Enabled** column, the toggle set to `Yes` indicates that your Datagen Source has started generating sample data.

1. Click **Commit & Deploy** at the upper right, enter a commit message, and click **Commit and Deploy**.
   This uses Cribl Stream's `git` integration to record the `default` Group's new configuration and deploy it to your Workers.



Input ID*: weblog
Enabled: Yes
`__inputId=='datagen:weblog'`
Datagen*

Data Generator File	Events Per Second Per Worker Node
weblog.log	10

Tags: Enter tags

Source (Datagen) configuration

If you'd like to verify that the Datagen is sending events, wait about a minute for them to start accumulating. Then, on the **Manage Sources / Datagen** page, click the **Live** button beside your configured Source. On the resulting **Live Data** tab, you should see events arrive within the default 10-second capture.

Add a Destination

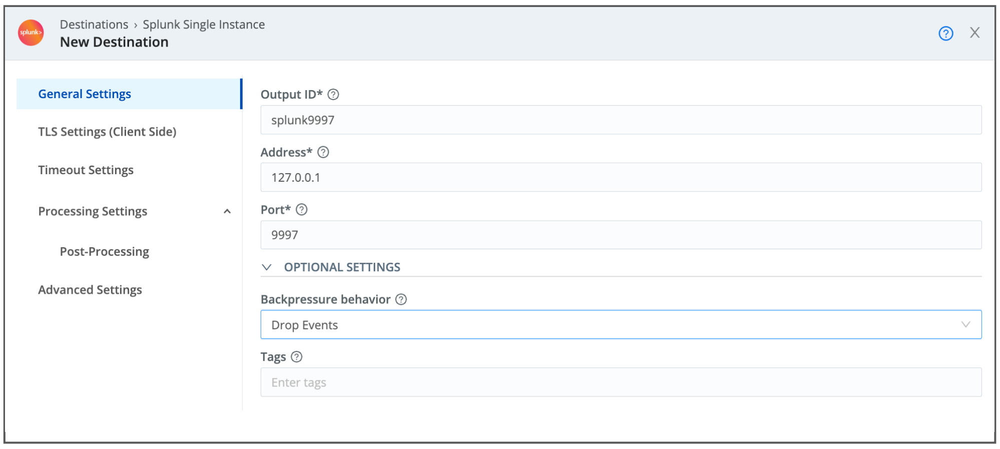
On the output side, choose any of these options:

- To configure a realistic output, but with no real dependencies and no license requirement, follow the Simulated Splunk Destination instructions just below.
- For a simpler option, jump to the Internal Destination instructions further down.
- To send data to a real receiver that you've already set up, see our Destinations topic for a link to the appropriate instructions.

Simulated Splunk Destination

Here, you'll go through the steps of configuring a typical Splunk Destination, but you'll use netcat to spoof the receiver on port 9997.

1. In the Cribl Stream Leader's UI, configure a Splunk Single Instance Destination, following these instructions. For this simulated output:
   - Set the **Address** to `127.0.0.1` (i.e., localhost).
   - Do not precede this IP address with any `http://` or `https://`
   - Leave the **Port** at the default `9997`.
   - Set the **Backpressure behavior** to `Drop Events`.



Destinations › Splunk Single Instance
New Destination

Output ID*: splunk9997
Address*: 127.0.0.1
Port*: 9997
OPTIONAL SETTINGS
Backpressure behavior: Drop Events
Tags: Enter tags

A glorious spoofed Splunk Destination config

2. Click **Commit** at the upper right, enter a commit message, and confirm the commit.
3. Click **Deploy** at the upper right to deploy this new configuration to your Workers.
4. In the first Worker instance's terminal/console, shell in, then enter `cd /opt/cribl/bin` to access Cribl Stream's CLI.
5. Enter `nc -h` to check whether netcat is installed on this Linux instance.

If the command fails, follow your Linux distro's steps for installing netcat. (E.g., for Ubuntu instructions, see Docker Notes below.)

6. Enter `./cribl nc -l -p 9997` to have netcat listen on port 9997, but simply discard data.

Docker Notes

If you're using a container like Docker, before shelling in at step 4 above, you'll need to first open a shell inside that container: `docker exec -it <CONTAINER ID> /bin/bash`

> In the above Docker Containers deployment example, you'd want to open the shell on the `docker_workers_1` container, whose `<CONTAINER ID>` was `a3de9ea8f46f`.
>
> Cribl's Docker containers currently run Ubuntu 20.04.6. You can install netcat with this sequence of commands:
>
> ```
> apt update
> apt install netcat
> ```

The data you'll now see displayed in the terminal will be gibberish, because of Splunk's proprietary data format.

If data isn't flowing, you might need to restart Workers. You can do this through Cribl Stream's UI. With Docker containers, use `docker-compose down`, followed by `docker-compose up`.

> You can streamline future Commit and Deploy steps by entering a **Default Commit Message**, and by collapsing actions to a combined **Commit and Deploy** button. Both options are available at **Settings** > **Global Settings** > **System** > **Git Settings** > **General**.

Internal Destination

As an alternative to the Splunk instructions above, you can configure Cribl Stream's built-in DevNull Destination to capture events and discard them. (This is the same Destination used in our single-instance Getting Started Guide tutorial.)

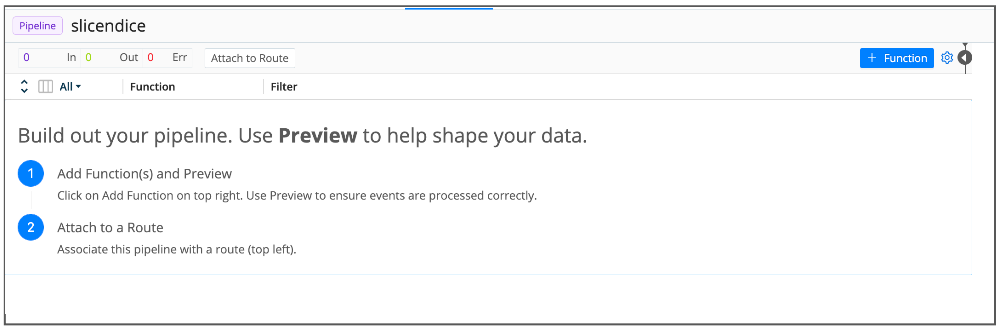
1. In the Leader UI's top nav, click **Manage** > **Groups**.
2. From the resulting **Groups** tab, click the Group name.
   (Working with a free license, we're implicitly configuring this Source on the `default` Group.)
3. From the top nav, select **Data** > **Destinations.**
4. Select **DevNull** from the **Data Destinations** page's tiles or left menu.
   (You can use the search box to jump to the **DevNull** tile.)
5. On the resulting **devnull** row, look for the **Live** indicator under **Enabled**. This confirms that the **DevNull** Destination is ready to accept events.
6. From the **Data Destinations** page's left nav, select the **Default** Destination at the top.
7. On the resulting **Manage Default Destination** page, verify that the **Default Output ID** drop-down points to the **devnull** Destination we just examined.
8. Click **Commit** at the upper right, enter a commit message, and confirm the commit.
9. Click **Deploy** at the upper right to deploy this new configuration to your Workers.

Add a Pipeline

To complete this distributed deployment with realistic infrastructure, let's set up a Pipeline and Route.

If you've already done the Getting Started Guide, you've already created a `slicendice` Pipeline. In the following steps, skip ahead to adding an Eval Function.

1. From the current Group's submenu, select **Processing** > **Pipelines**.
2. At the **Pipelines** pane's upper right, click **Add Pipeline**, then select **Create Pipeline**.
3. In the new Pipeline's **ID** field, enter a unique identifier (e.g., `slicendice`).
4. Optionally, enter a **Description ** of this Pipeline's purpose.
5. Click **Save**.



Build out your pipeline. Use **Preview** to help shape your data.

1 Add Function(s) and Preview
Click on Add Function on top right. Use Preview to ensure events are processed correctly.

2 Attach to a Route
Associate this pipeline with a route (top left).

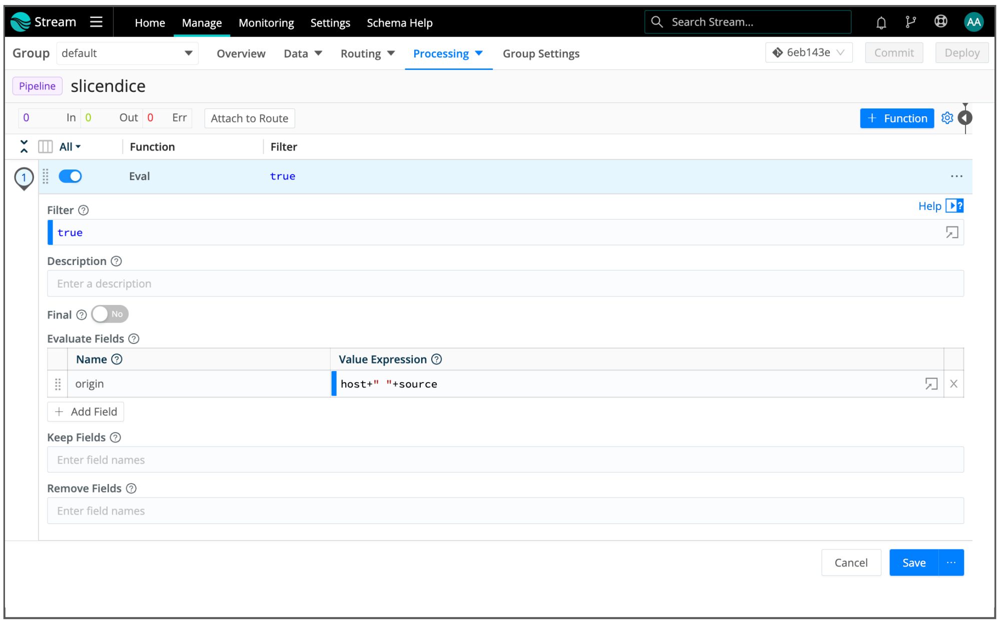
Pipeline saved

Add an Eval Function

Now add an Eval Function to the Pipeline:

1. At the **Pipelines** pane's upper right, click **Add Function**.
2. Search for the `Eval` Function, and click its link to add it.
3. In the new Function's **Evaluate Fields** section, click **Add Field**.
4. In the new row's **Name** column, name the field `origin`.
5. In the new row's **Value Expression** column, enter: `host+" "+source`

This new `origin` field will concatenate the host and source fields from incoming events.



Group: default

Pipeline slicendice

0 In 0 Out 0 Err

	Function	Filter
1	Eval	true

Filter: true

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
origin	host+" "+source

Keep Fields: Enter field names

Remove Fields: Enter field names

Adding Eval Function to Pipeline

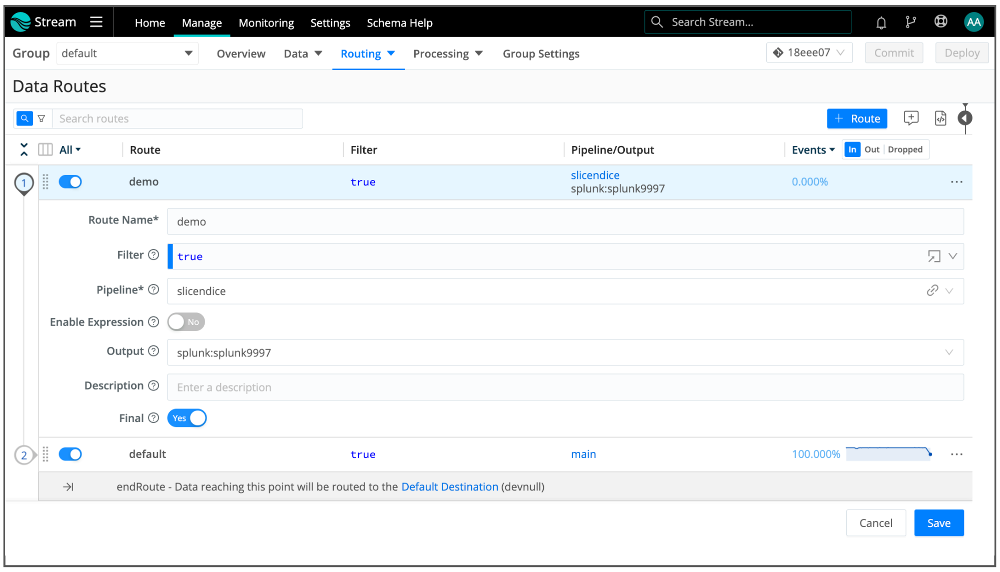
6. Click **Save** to store the Function's configuration.
7. **Commit** and **Deploy** the new Pipeline configuration.
8. Optionally, open the right pane and click **Capture Data** to verify throughput.

Add a Route

If you've already done the Getting Started Guide, you've already created a `demo` Route, attached to the `slicendice` Pipeline. In the following steps, just modify the Route to send data to the new Destination you configured above.

1. At the **Pipelines** page's top left, click **Attach to Route**.
   This displays the **Data Routes** page.
2. Click `Add Route`.
3. Enter a unique **Route Name**, like `demo`.
4. Leave the **Filter** field set to its `true` default, allowing it to deliver all events.
5. Set the **Pipeline** drop-down to our configured `slicendice` Pipeline.
6. Set the **Output** drop-down to the Destination you configured above. If you boldly chose the Simulated Splunk Destination, this will be named something like `splunk:splunk9997`.
7. You can leave the **Description** empty, and leave **Final** set to `Yes`.
8. Grab the new Route by its left handle, and drag it to the top of the Routing table, so that our new Route will process events first. You should see something like the screenshot below.
9. Click **Save** to save the new Route configuration.
10. **Commit** and **Deploy** your changes.

11. Still assuming you configured a simulated Splunk output, look at the terminal where you're running netcat. You should now see events arriving.



Route	Filter	Pipeline/Output	Events
1 demo	true	slicendice splunk:splunk9997	0.000%
2 default	true	main	100.000%

Route Name*: demo
Filter: true
Pipeline*: slicendice
Enable Expression: No
Output: splunk:splunk9997
Description: Enter a description
Final: Yes

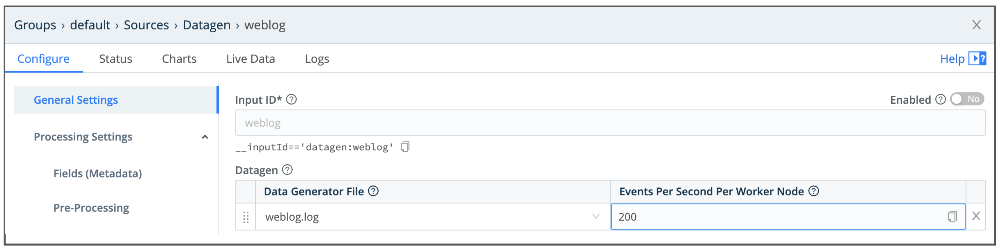
endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Route attached to Pipeline and Destination

Managing Workers (Scaling)

With all our infrastructure in place, let's look at how a Cribl Stream distributed deployment scales up to balance the incoming event load among multiple Workers.

1. In the Leader UI's top nav, click **Manage** > **Groups**.
2. From the Groups page, click the Group name.
   (Working with a free license, we're implicitly configuring this Source on the `default` Group.)
3. From the top nav, select **Manage** > **Data** > **Sources**, and find the **Datagen** Source you configured earlier.
4. Click this Datagen's row to reopen its config modal.
5. Reset the **Events per Second per Worker Node** from the default `10` EPS to a high number, like `200` EPS.



Input ID*: weblog
Enabled: No
`__inputId=='datagen:weblog'`
Datagen

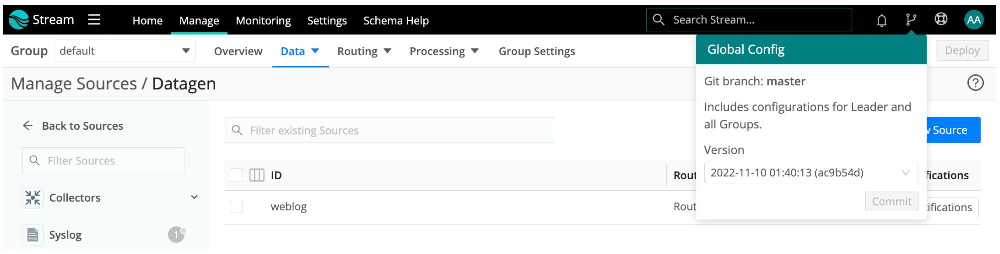
Data Generator File	Events Per Second Per Worker Node
weblog.log	200

Flooding events into Cribl Stream

6. Click **Save**, then **Commit** and **Deploy** this higher event load.

7. From the Leader's top nav, also select **Global Config** > **Commit** to commit the Leader's newest config version. In the resulting modal, click **Commit** again to confirm.

This uses Cribl Stream's `git` integration to save a global configuration point for your whole deployment, which you can roll back to.



Manage Sources / Datagen

Global Config

Git branch: master

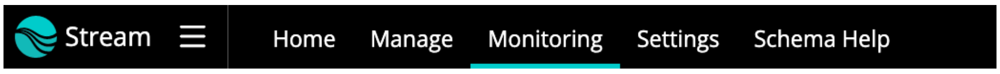
Includes configurations for Leader and all Groups.

Version

2022-11-10 01:40:13 (ac9b54d)

Committing Leader's config

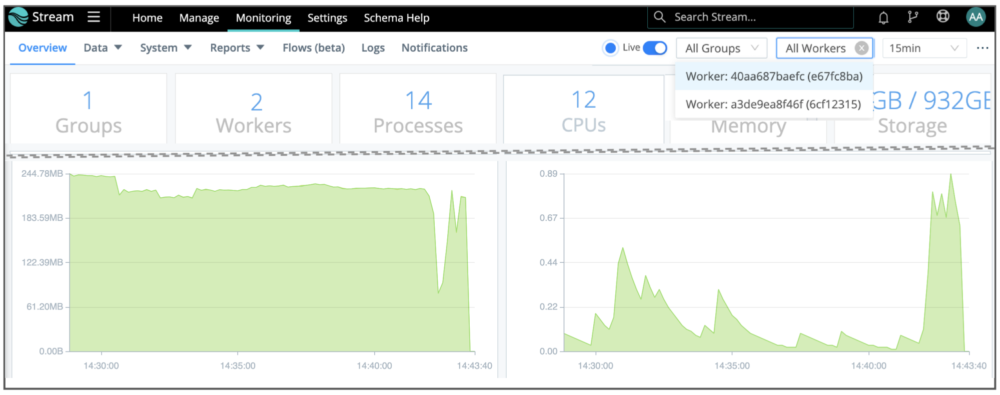
8. From the Leader's top nav, click **Monitoring**.



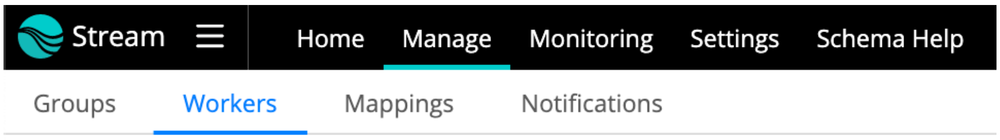
Monitoring tab

9. On the resulting **Monitoring** page, watch as the following changes unspool over the next few minutes:

- The **CPU Load Average (1 min)** will spike as the higher event volume floods the system.
- The **Workers** indicator at upper left will drop to `0` as the Workers restart with the new configuration you've deployed to them, and then rebound to `2`.
- If you use the **All Workers** drop-down at upper right to toggle between your two individual Workers, you should see that Cribl Stream is balancing roughly equal event loads among them.



10. To confirm both Workers' `alive` status: Click **Manage**, then click the **Workers** tab.



Workers tab

11. To wrap up, repeat this procedure's first six steps to set the Datagen's rate back down to `10` EPS, and then save, commit, and deploy all changes.

What Have We Done?

This completes your setup of a basic distributed deployment, using a free license, and configuring a single Worker Group of two identically configured Worker Processes. (You can extrapolate these same techniques you've just mastered to spin up virtually any number of Workers in the same way.)

To see how you can set up multiple Worker Groups – with separate configurations optimized for separate data flows – continue to the next section, noting its licensing prerequisites.

If you're deferring or skipping that option, jump ahead to Cleaning Up.

Add and Manage Multiple Worker Groups (Optional)

To add and manage more than one Worker Group – everything in this optional section – you'll need an Enterprise or Sales Trial license for your on-prem Cribl Stream deployment, or an Enterprise (or equivalent) plan for your Cribl.Cloud Organization. For details on adding multiple Worker Groups on Cribl.Cloud, see Cribl.Cloud Worker Groups.

> See Licensing for how to acquire and install one of the above license types. Install the license on your Leader instance, and then commit this as a Leader config change (top nav's **Global Config** dialog), before you proceed.

Here, we'll build on the infrastructure we've created so far to:

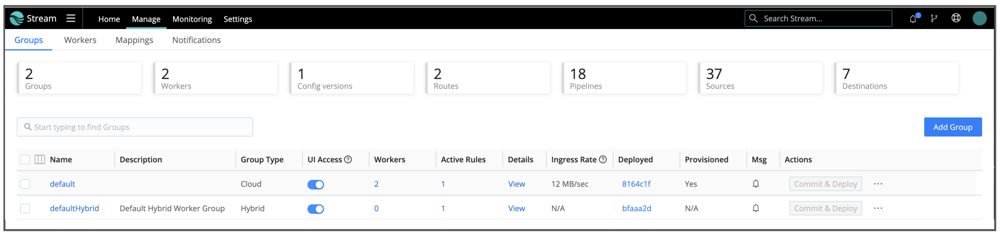
- Configure Mapping Rules.
- Verify how Cribl Stream balances large data volumes among Worker Processes.
- Add a second Worker Group, data Source, and Destination.
- Add a second Pipeline and attach it to its own Route.
- Reconfigure Mapping Rules to send each Source's data through a separate Group.

To keep this Quick Start tutorial focused on techniques, rather than on configuring lots of infrastructure, we'll assign just one Worker to each Worker Group – one of the two Workers we launched above. But in production, you'll be able to apply the same principles to setting up any number of Worker Groups, with any number of Workers.

Multiple-Group Setup

With an Enterprise or Sales Trial license/plan, Cribl Stream's UI adds some extra features.

The **Manage** > **Groups** page, shown here, now supports multiple Groups. (For now, you still see only the single `default` Group – but we'll change that a few sections down.)



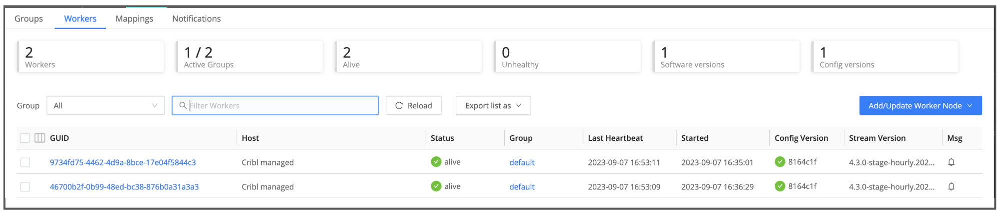
Name	Description	Group Type	UI Access	Workers	Active Rules	Details	Ingress Rate	Deployed	Provisioned	Msg	Actions
default		Cloud		2	1	View	12 MB/sec	8164c1f	Yes		Commit & Deploy
defaultHybrid	Default Hybrid Worker Group	Hybrid		0	1	View	N/A	bfaaa2d	N/A		Commit & Deploy

UI for multiple Worker Groups

On Cribl.Cloud, the **Groups** page adds extra columns. You'll see each Group's Worker type (hybrid versus Cloud/Cribl-managed) and, for Cloud Groups, the anticipated ingress rate and Provisioned status. For details, see Cribl.Cloud Worker Groups.

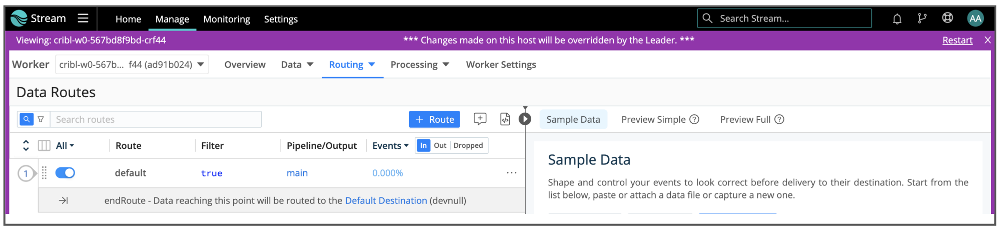
Note that if you've enabled Worker UI access, you can click directly through to each of your Workers. This feature will come in handy just below. To try it out:

1. Either click the link in the desired Group's **Workers** column, or click the submenu's **Workers** tab.
2. Either way, from the resulting **Workers** tab, click the desired Worker's link in the **GUID** column.
3. A purple header indicates that you're viewing a Worker's UI.
4. To return to the Leader's UI, just click the top nav's **Manage** option.)



GUID	Host	Status	Group	Last Heartbeat	Started	Config Version	Stream Version	Msg
9734fd75-4462-4d9a-8bce-17e04f5844c3	Cribl managed	✓ alive	default	2023-09-07 16:53:11	2023-09-07 16:35:01	✓ 8164c1f	4.3.0-stage-hourly.202...	
46700b2f-0b99-48ed-bc38-876b0a31a3a3	Cribl managed	✓ alive	default	2023-09-07 16:53:09	2023-09-07 16:36:29	✓ 8164c1f	4.3.0-stage-hourly.202...	

Select a Worker's GUID link to tunnel to that Worker



Viewing: cribl-w0-567bd8f9bd-crf44 *** Changes made on this host will be overridden by the Leader. ***

Data Routes

	Route	Filter	Pipeline/Output	Events
1	default	`true`	main	0.000%

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Sample Data

Shape and control your events to look correct before delivery to their destination. Start from the list below, paste or attach a data file or capture a new one.

Leader's remote view of Worker's UI

Check/Restart Workers

For the remaining steps, we want to make sure both our Workers (configured earlier) are up. Cribl Stream's top header should indicate `2 WORKERS`. You can verify that they're `alive` by clicking the top nav's **Manage** > **Workers** tabs, as shown earlier. If so, proceed to Map Groups by Config.

If either or both Workers are down, restart them:

1. Make sure you've enabled Worker UI access. (It's time!)
2. To click through to a dormant Worker's UI, use either of the left-nav options covered just above: either the **Workers** page, or the **Manage** > **Groups** > `default` submenu of individual Worker IDs.
3. When you see that Worker's UI (purple header), click **Restart** at the upper right.
4. Confirm your choice, and wait for a **Server has been restarted** message to appear for a few seconds.
5. Click the top nav's **Manage** option to return to the Leader's UI.
6. If the other Worker is down, repeat the above steps to restart it as well.

Map Groups by Config

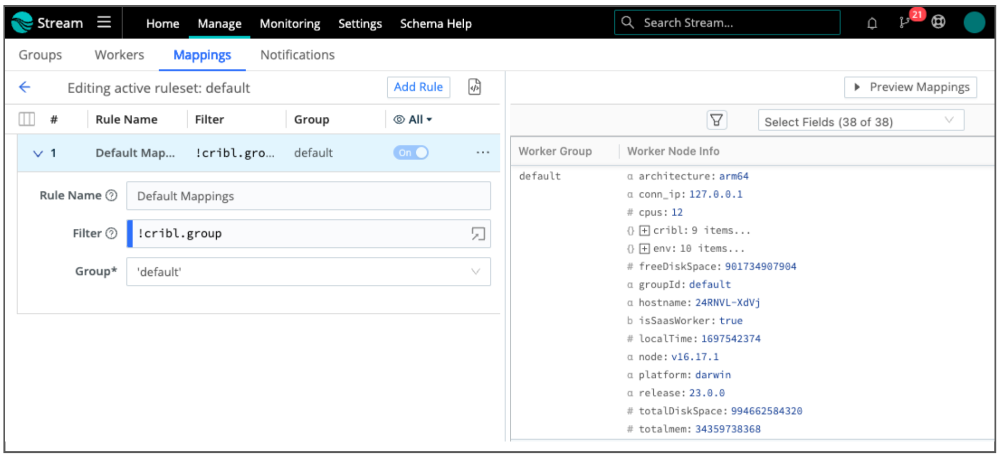
With both Workers confirmed up, let's look at how Cribl Stream has automatically mapped all these existing workers to the `default` Group.

1. From the Leader's top nav, click **Manage** > **Mappings**.
2. You now see a default Mapping Ruleset, also literally named `default`. Click it.

A Cribl Stream Leader can have multiple Mapping Rulesets configured, but only one can be active at a time.

3. This `default` Ruleset contains one initial Rule, literally named `Default Mappings`. Expand its accordion as shown below.

(The `default` Ruleset and Rule naming are separate from the `default` Group's naming. All of these out-of-the-box starting configurations have been named...literally.)



#	Rule Name	Filter	Group	All
1	Default Map...	!cribl.gro...	default	On

Rule Name: Default Mappings
Filter: !cribl.group
Group*: 'default'

Worker Group	Worker Node Info
default	architecture: arm64
	conn_ip: 127.0.0.1
	cpus: 12
	cribl: 9 items...
	env: 10 items...
	freeDiskSpace: 901734907904
	groupId: default
	hostname: 24RNVL-XdVj
	isSaasWorker: true
	localTime: 1697542374
	node: v16.17.1
	platform: darwin
	release: 23.0.0
	totalDiskSpace: 994662584320
	totalmem: 34359738368

Default Mapping Rule

Below the **Rule Name**, a Mapping Rule has two functional fields:

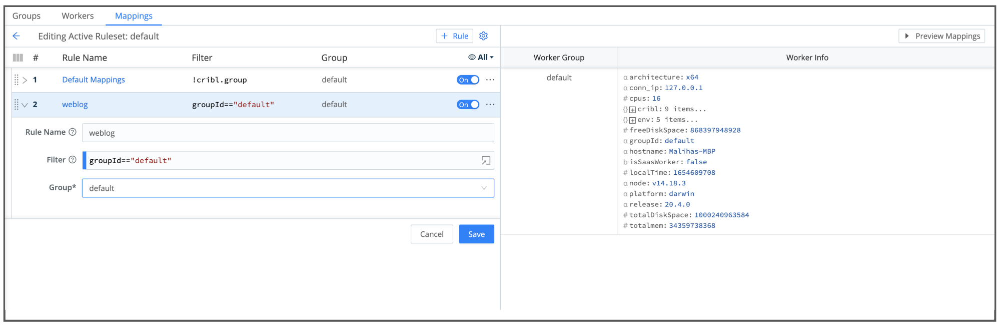
- **Filter**: `!cribl.group` – this value expression specifies “Not already assigned to a Worker Group.”
- **Group**: `default` – this value specifies “Assign to the `default` Group.”

So this is a catch-all rule. By following it, the Leader has assigned all (both) registered Workers to the `default` Group.

Let’s make a more-specific rule, mapping a specific Worker (by hostname) to this Group, which receives events from our `weblog` Datagen Source.

Add Mapping Rules

1. Click **Add Rule** at the upper right. Then configure the new Rule as shown below:
   - **Rule Name**: `weblog` – for simplicity.
   - **Filter**: `platform=="<this-worker's-platform>"` – get this value from the right Preview pane. Look for the `platform` field’s value. In the example shown below, we got `"darwin"`.
   - **Group**: `default` – this is the only option available.



#	Rule Name	Filter	Group	All
1	Default Mappings	!cribl.group	default	On
2	weblog	groupId=="default"	default	On

Rule Name: weblog
Filter: groupId=="default"
Group*: default

Worker Group	Worker Info
default	architecture: x64
	conn_ip: 127.0.0.1
	cpus: 16
	cribl: 9 items...
	env: 5 items...
	freeDiskSpace: 868397948928
	groupId: default
	hostname: Malihas-MBP
	isSaasWorker: false
	localTime: 1654609708
	node: v14.18.3
	platform: darwin
	release: 20.4.0
	totalDiskSpace: 1000240963584
	totalmem: 34359738368

A specific Mapping Rule by Filter condition

2. Click **Save** to add this Rule.

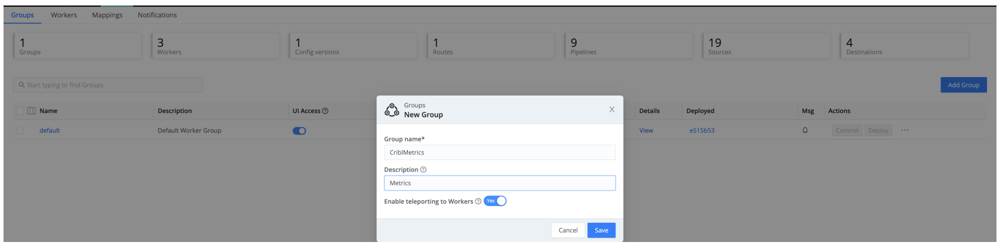
3. Confirm the warning that changes will take effect immediately.
4. From the top nav, select **Global Config** > **Commit** to commit the Leader's new config.

For more Mapping Rules/Rulesets details and examples, see Distributed Deployment.

Next, we'll add a second Worker Group; add a second Source (relaying Cribl Stream's internal metrics); and then add another Mapping Rule, to map our second Worker to the new Group.

Add Another Worker Group

1. From the Leader's top nav, click **Manage** > **Groups**.
2. On the resulting Groups page, click **New Group**.
3. Name the new Group `CriblMetrics` to match its purpose.
4. Toggle **UI access** to `Yes`.

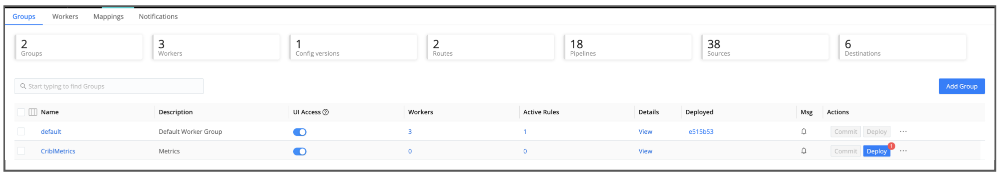


Name	Description	UI Access	Details	Deployed	Msg	Actions
default	Default Worker Group	●	View	e515b53		Commit Deploy ···

Group name*: CriblMetrics
Description: Metrics
Enable teleporting to Workers: Yes

Creating a new Group

4. Save the Group.
5. Click **Deploy** on the new Group's row, and confirm your choice. The new Group should deploy immediately.



Name	Description	UI Access	Workers	Active Rules	Details	Deployed	Msg	Actions
default	Default Worker Group	●	3	1	View	e515b53		Commit Deploy ···
CriblMetrics	Metrics	●	0	0	View			Commit Deploy ···

Second Groups saved, ready to Deploy

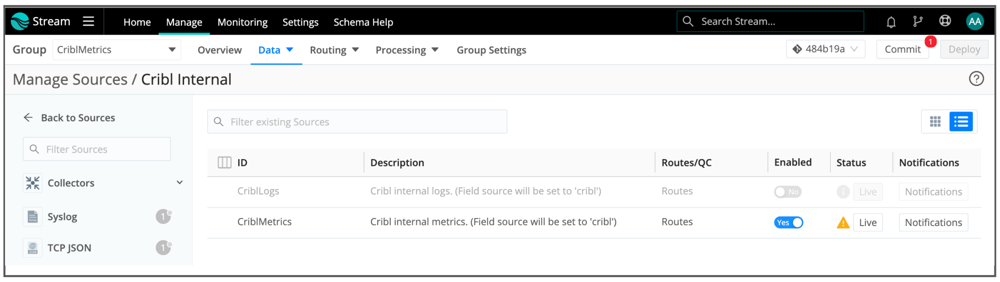
6. From the top nav, select **Global Config** > **Commit** to commit the new config on the Leader.

Add Another Source

On the new Group, we'll now enable a **Cribl Internal: Metrics** Source, representing a second data type.

1. From the Leader's top nav, click **Manage** > **Groups** > `CriblMetrics`.
2. From the resulting submenu, click **Data** > **Sources**.

3. From the **Data Sources** page's tiles or left menu, select **Cribl Internal**.
4. On the **Manage Sources / Cribl Internal** page, toggle **Enable** to `Yes` in the `CriblMetrics` row.
5. Confirm that you want to enable `CriblMetrics`.



ID	Description	Routes/QC	Enabled	Status	Notifications
CriblLogs	Cribl internal logs. (Field source will be set to 'cribl')	Routes	No	Live	Notifications
CriblMetrics	Cribl internal metrics. (Field source will be set to 'cribl')	Routes	Yes	Live	Notifications

CriblMetrics Source enabled

6. From the top nav, select **Global Config** > **Commit** to commit the new config on the Leader and Group.
7. Click **Deploy** at the upper right to deploy this new configuration to your Workers.

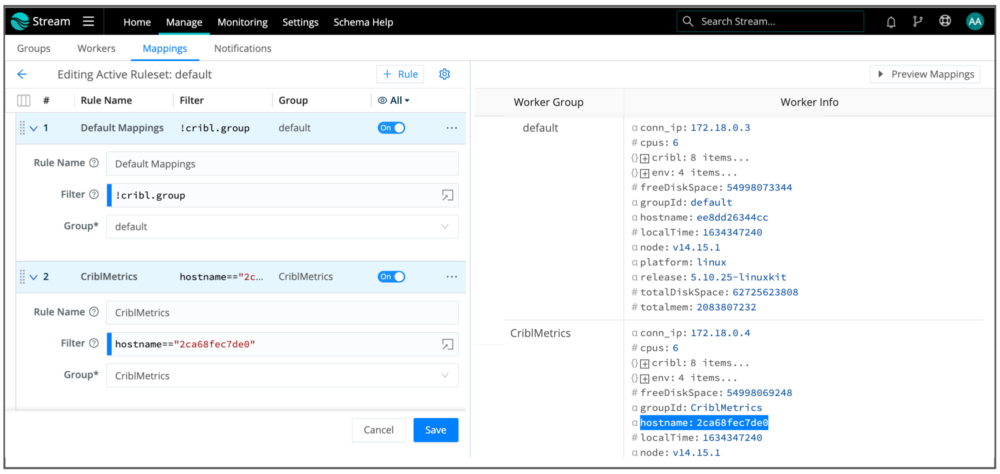
Map Workers to Groups

Now let's map this new group to the CriblMetrics Source's incoming events:

1. From the Leader's top nav, click **Manage** > **Groups**.
2. From the Groups page, click the **Mappings** tab.
3. Click the `default` Mapping Ruleset to open it.
4. Click **Add Rule** at the upper right. Then configure the new Rule as shown below:

- **Rule Name**: `CriblMetrics` – for simplicity.
- **Filter**: `hostname=="<this-worker's-hostname>"` – get this value from the right Preview pane. In the second Worker down, look for the `hostname` field's value. In the example shown below, we got `"2ca68fec7de0"`.
- **Group**: `CriblMetrics` – this is the Group we want to map.

5. Move the `Default Mappings` rule to the bottom of the Ruleset, reflecting its catch-all function.
6. Click **Save** to store the new configuration.
7. Confirm the warning that changes will take effect immediately.



Editing Active Ruleset: default

#	Rule Name	Filter	Group	
1	Default Mappings	`!cribl.group`	default	On
2	CriblMetrics	`hostname=="2c...`	CriblMetrics	On

Rule 1 — Rule Name: Default Mappings; Filter: `!cribl.group`; Group*: default

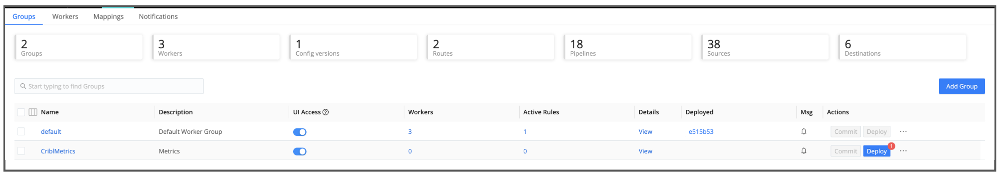
Rule 2 — Rule Name: CriblMetrics; Filter: `hostname=="2ca68fec7de0"`; Group*: CriblMetrics

Worker Group	Worker Info
default	conn_ip: 172.18.0.3 cpus: 6 cribl: 8 items... env: 4 items... freeDiskSpace: 54998073344 groupId: default hostname: ee8dd26344cc localTime: 1634347240 node: v14.15.1 platform: linux release: 5.10.25-linuxkit totalDiskSpace: 62725623808 totalmem: 2083807232
CriblMetrics	conn_ip: 172.18.0.4 cpus: 6 cribl: 8 items... env: 4 items... freeDiskSpace: 54998069248 groupId: CriblMetrics hostname: 2ca68fec7de0 localTime: 1634347240 node: v14.15.1

Adding a Rule to map the CriblMetrics Worker to its own Group

8. From the top nav, select **Global Config** > **Commit** to commit the Leader's new config.

We now have two data Sources and two Worker Groups – one each for (Web) logs versus (Cribl Internal) metrics – along with two Mapping Rules to map data accordingly. To confirm the Workers' assignment to the two Groups, click the top nav's **Manage** tab, then select **Groups** from the submenu:

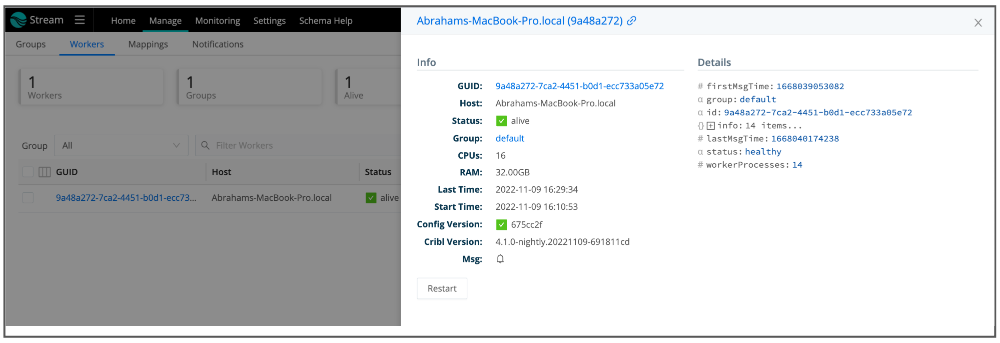


2 Groups | 3 Workers | 1 Config versions | 2 Routes | 18 Pipelines | 38 Sources | 6 Destinations

Name	Description	UI Access	Workers	Active Rules	Details	Deployed	Msg	Actions
default	Default Worker Group	On	3	1	View	e515b53		Commit Deploy
CriblMetrics	Metrics	On	0	0	View			Commit Deploy

Manage Groups page confirms Workers' mapping and assignment

To confirm further details about the Workers, click the **Workers** tab, and on the resulting **Workers** page, click anywhere on the Worker Node's row to reveal more details:



Abrahams-MacBook-Pro.local (9a48a272)

Info

GUID:	9a48a272-7ca2-4451-b0d1-ecc733a05e72
Host:	Abrahams-MacBook-Pro.local
Status:	✓ alive
Group:	default
CPUs:	16
RAM:	32.00GB
Last Time:	2022-11-09 16:29:34
Start Time:	2022-11-09 16:10:53
Config Version:	✓ 675cc2f
Cribl Version:	4.1.0-nightly.20221109-691811cd

Details

```
firstMsgTime: 1668039053082
group: default
id: 9a48a272-7ca2-4451-b0d1-ecc733a05e72
info: 14 items...
lastMsgTime: 1668040174238
status: healthy
workerProcesses: 14
```

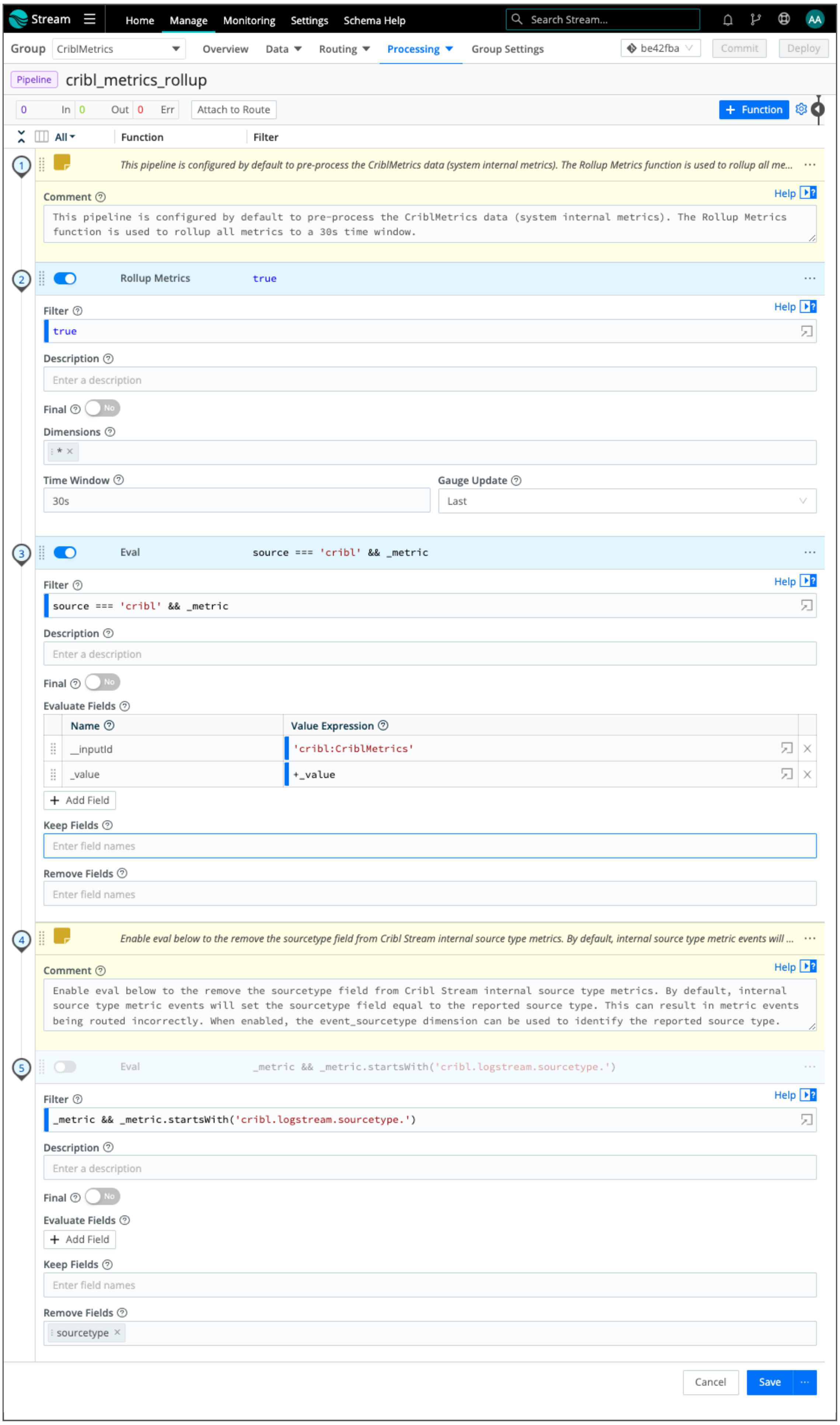
Worker Node Details

Configure Metrics Output

With incoming metrics now mapped to our second Worker Group, we next need to configure this Group's output. Here, we'll rely on a metrics-oriented Pipeline and a Destination that ship with Cribl Stream, and create a new Route to connect everything up.

Examine the Metrics Pipeline

1. From the Leader's top nav, click **Manage** > **Groups** > `CriblMetrics`.
2. From the resulting submenu, select **Processing** > **Pipelines**.
3. On the **Pipelines** page, find the `cribl_metrics_rollup` Pipeline, and click it to expand it.
4. Expand this Pipeline's Functions (including Comments) to see its configuration. It's preconfigured with a Rollup Metrics Function to aggregate metrics to a 30-second time window. Next is an Eval Function that filters for Cribl (Cribl Stream) internal metrics and tags them on outgoing events with a new field.



cribl_metrics_rollup

1 *This pipeline is configured by default to pre-process the CriblMetrics data (system internal metrics). The Rollup Metrics function is used to rollup all me...*

Comment

```
This pipeline is configured by default to pre-process the CriblMetrics data (system internal metrics). The Rollup Metrics
function is used to rollup all metrics to a 30s time window.
```

2 Rollup Metrics `true`

Filter: `true`

Description: Enter a description

Final: No

Dimensions: `*`

Time Window: 30s

Gauge Update: Last

3 Eval `source === 'cribl' && _metric`

Filter: `source === 'cribl' && _metric`

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
__inputId	`'cribl:CriblMetrics'`
_value	`+_value`

+ Add Field

Keep Fields: Enter field names

Remove Fields: Enter field names

4 *Enable eval below to the remove the sourcetype field from Cribl Stream internal source type metrics. By default, internal source type metric events will ...*

Comment

```
Enable eval below to the remove the sourcetype field from Cribl Stream internal source type metrics. By default, internal
source type metric events will set the sourcetype field equal to the reported source type. This can result in metric events
being routed incorrectly. When enabled, the event_sourcetype dimension can be used to identify the reported source type.
```

5 Eval `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

Filter: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

Description: Enter a description

Final: No

Evaluate Fields

+ Add Field

Keep Fields: Enter field names

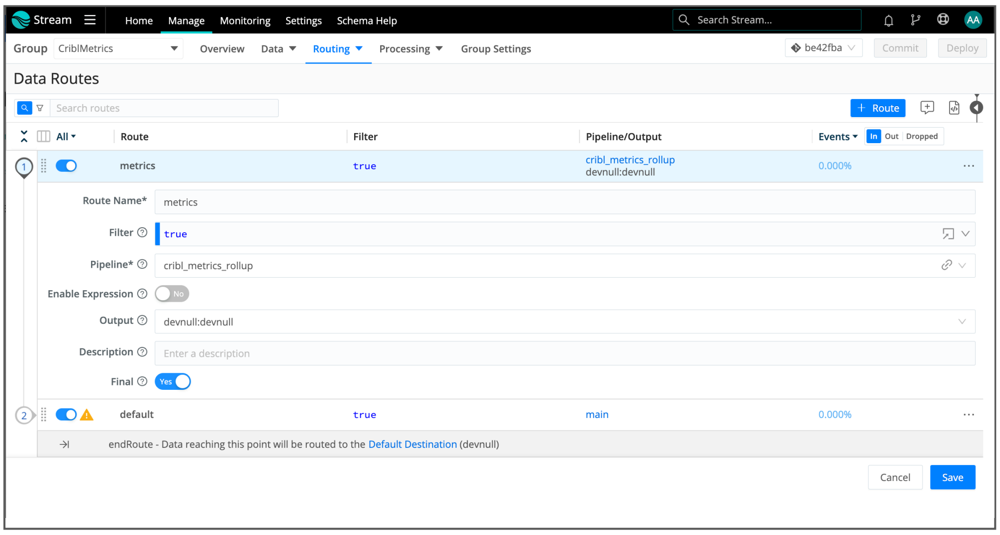
Remove Fields: sourcetype

cribl_metrics_rollup Pipeline, shipped with Cribl Stream

Add Another Route

We'll connect this existing Pipeline to a new Route:

1. At the **Pipelines** page's top left, click **Attach to Route**.
   This displays the **Data Routes** page.
2. Click `Add Route`.
3. Enter a unique **Route Name**, like `metrics`.
4. Leave the **Filter** field set to its `true` default, allowing it to deliver all events.
5. Make sure the **Pipeline** drop-down is set to `cribl_metrics_rollup`.
6. As the **Output** (Destination), select our old friend `devnull:devnull`.
   This is Cribl Stream's preconfigured Destination that simulates a downstream service while simply dropping events.
7. You can leave the **Description** empty, and leave **Final** set to `Yes`.
8. Grab the new Route by its handle, and drag it to the top of the Routing table, so that our new Route will process events first. You should see something like the screenshot below.
9. Click **Save** to save the new Route configuration.
10. **Commit** and **Deploy** your changes.



Route	Filter	Pipeline/Output	Events
1 metrics	true	cribl_metrics_rollup devnull:devnull	0.000%
2 default	true	main	0.000%

Route Name*: metrics
Filter: true
Pipeline*: cribl_metrics_rollup
Enable Expression: No
Output: devnull:devnull
Description: Enter a description
Final: Yes

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

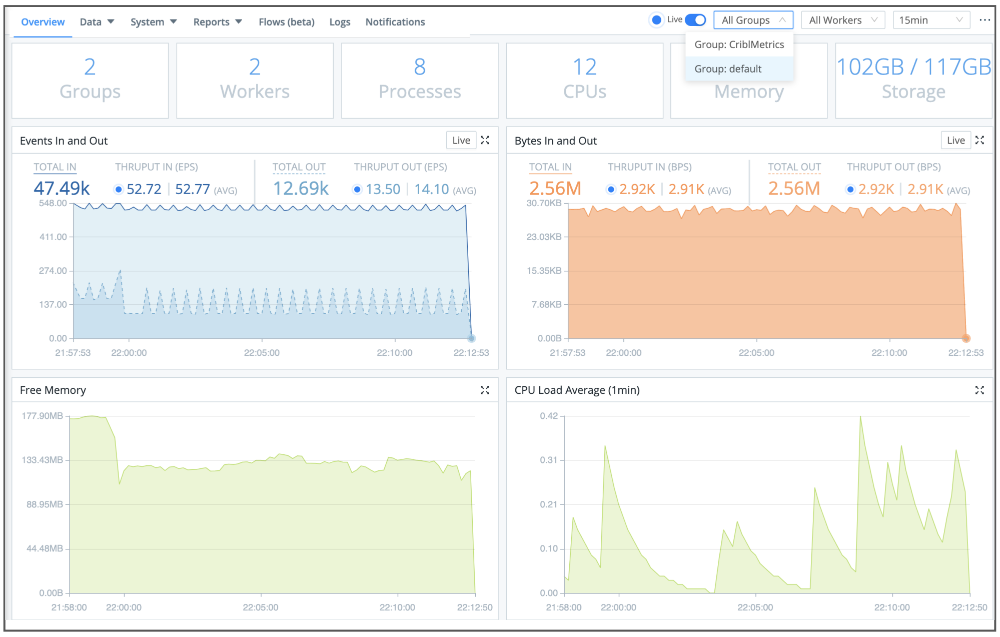
Cribl metrics Route, Pipeline, and Destination wired up

Verify the Multi-Group Deployment

From the sparkline on the Route you just configured (see the screenshot above), you can already see that metrics data is flowing all the way "out" of Cribl Stream – simulated here by the DevNull Destination.

To verify the whole configuration you've created, click the top nav's **Monitoring** tab. On the **Monitoring** page, toggle the **All Groups** drop-down (upper right), toggle between the two Worker Groups to see the division of labor:

- Group `default` (the out-of-the-box Group we configured first) handles the `weblog.log` data.
- Group `CriblMetrics` handles the metrics data.

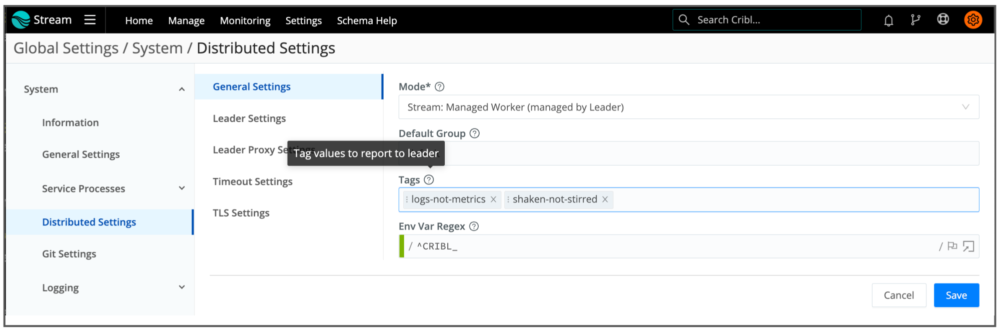


Monitoring individual Worker Groups' throughput

All the Distributed Things – Review

Before a final section where you can tear down your infrastructure, here's a recap of the simple (but expandable) distributed model we've created, with some ideas for expanding it:

- A distributed deployment enables Cribl Stream to scale out to higher data volumes. This load-balancing occurs even with a single Worker Group, in which all Workers share the same configuration.
- By adding multiple Worker Groups, you can partition Worker Nodes (and their data flows) by different configurations. In this demonstration, we simply mapped Workers to Groups by the Workers' `hostname`. But you can map by a range of arbitrary criteria to meet your production needs.
- E.g.: Different Groups can be managed by different teams. You can filter on DNS rules to send relevant data to the relevant team's Group.
- Different Groups can also maintain configurations for different regions' data privacy or data retention requirements.
- You can also Map workers arbitrarily using Tags and other options.



Mode*: Stream: Managed Worker (managed by Leader)

Default Group:

Tags: logs-not-metrics, shaken-not-stirred

Env Var Regex: / ^CRIBL_ /

Setting arbitrary Tags on a managed Worker

More About Worker Groups, Nodes, and Processes – Definitions

Cribl Stream refers to "Workers" at several levels. Now that you've been initiated into building distributed deployments, here's a guide to the fine points of distinguishing these levels:

- A **Worker Group** holds one or multiple **Worker Nodes**.
- Each Worker Node functions like an independent Cribl Stream single-instance deployment. Worker Nodes don't coordinate directly with each other – they're coordinated only through communication to/from the Leader Node.
- Each Worker Node contains a configured number of **Worker Processes**. Unlike the above grouping abstractions, this is the level that actually processes data. To load-balance among Worker Processes, Cribl Stream's API Process round-robins incoming connections to them.

When deploying on AWS/EKS, Worker Groups should not span Availability Zones. If you have EBS persistent volumes, and a node fails, its replacement won't be able to access the peer volume across AZs.

Cleaning Up

If and when you choose to shut down the infrastructure you've configured for this demonstration:

- Navigate to Cribl Stream's `default` Group > **Data** > **Sources** > **Datagen**, and switch off the toggle beside `weblog.log`.
- If you configured a second Worker Group: Navigate to Cribl Stream's `CriblMetrics` Group > **Data** > **Sources** > **Cribl Internal**, and disable the toggle beside **CriblMetrics**.
- If you have `netcat` running on a Worker's terminal/console, `^C` it.
- There's no need (or way) to switch off the **DevNull** Destination – it just is.

- If desired, **Commit** and **Deploy** these changes.
- If you’re running the Cribl Stream server(s) on cloud instances that will (ultimately) incur charges, you’re now free to shut down those cloud resources.

Next Steps

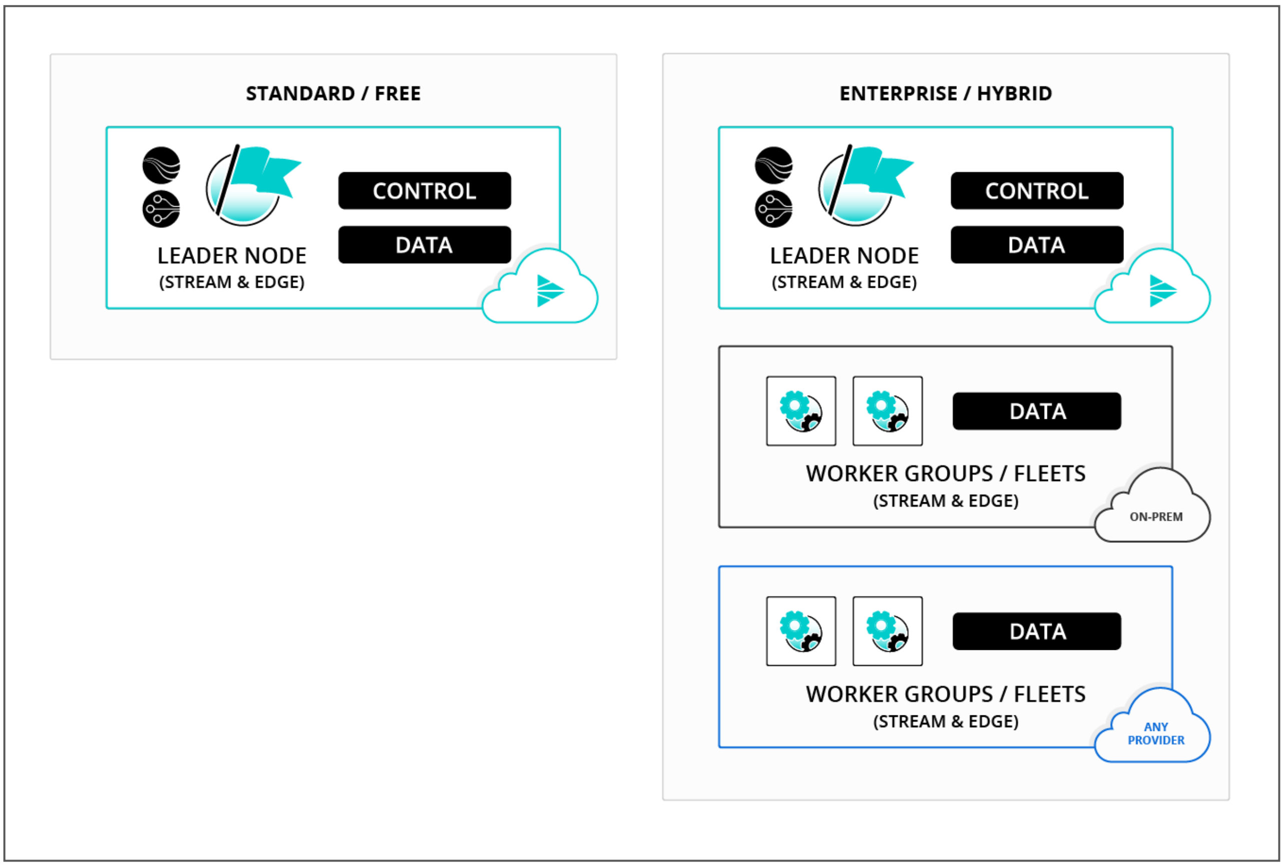
Interested in guided walk-throughs of more-advanced Cribl Stream features? We suggest that you next check out these further resources.

- Distributed Deployment: All the details behind the deployment approach you just mastered in this tutorial.
- Cribl Stream Sandboxes: Work through general and specific scenarios in a free, hosted environment, with terminal access and real data inputs and outputs.
- Use Cases documentation: Bring your own services to build solutions to specific challenges.
- Cribl Concept: Pipelines – Video showing how to build and use Pipelines at multiple Cribl Stream stages.
- Cribl Concept: Routing – Video about using Routes to send different data through different paths.

2. Cribl.Cloud

The fast alternative to downloading and self-hosting Cribl Stream software is to launch Cribl.Cloud. This SaaS version, which comes in both a free and a paid version, places the Leader and the Worker 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?

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

- Full Cribl Stream 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.
- Access to Cribl Search to search, explore, and analyze machine data in place.
- 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.
- Quick expansion of your Cribl.Cloud deployment beyond the free tier's limits by purchasing credits toward metered billing. Pay only for what you use.

If you're new to Cribl Stream, please see our Basic Concepts page and Getting Started Guide for orientation. This Cribl.Cloud documentation focuses on a Cloud deployment's differences from other deployment options – referred to as "customer-managed deployments."

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

Cloud Pricing

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

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

2.1. Initial Cribl.Cloud Setup

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

Your Organization will display a dedicated portal, 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.

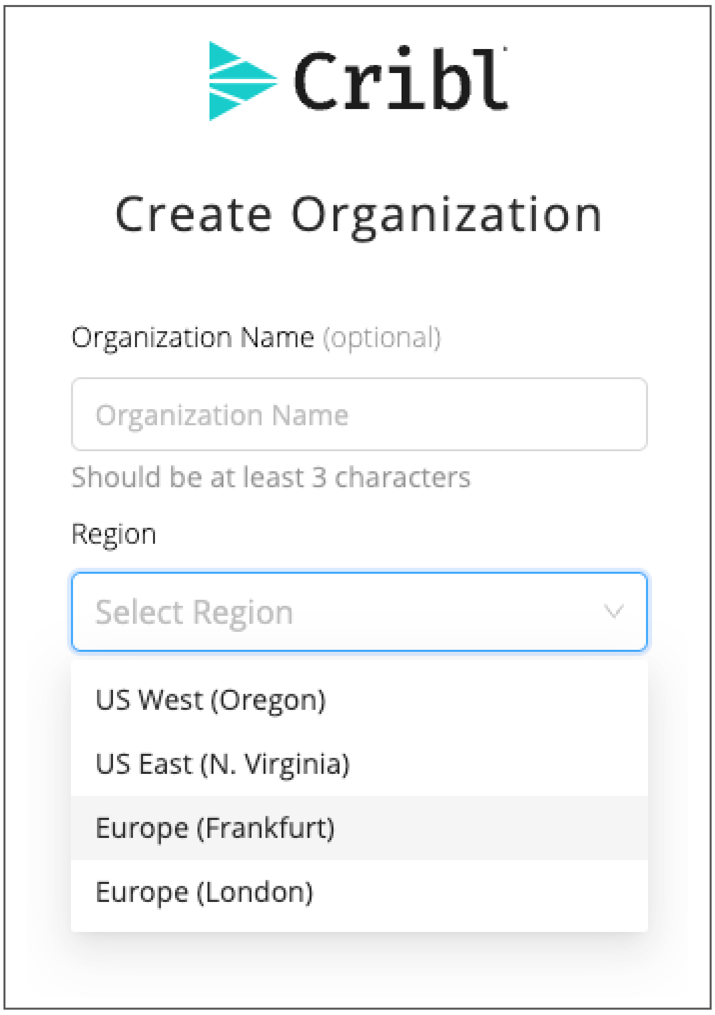
> The Cribl.Cloud Suite is listed on AWS Marketplace. When you're ready for a paid plan, you can use your Enterprise Discount Program (EDP) credits here to run Cribl products, billed through your AWS account – with no need for a separate procurement process.
>
> Cribl is SOC 2 (Service Organization Control 2) Type II security compliance certified.

Registering a Cribl.Cloud Portal

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

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

1. Start at: https://cribl.cloud/signup/
2. Register with your work email address.
3. 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 the following Regions:
    - `US West (Oregon)`
    - `US East (Virginia)`
    - `Europe (Frankfurt)`
    - `Europe (London)`
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

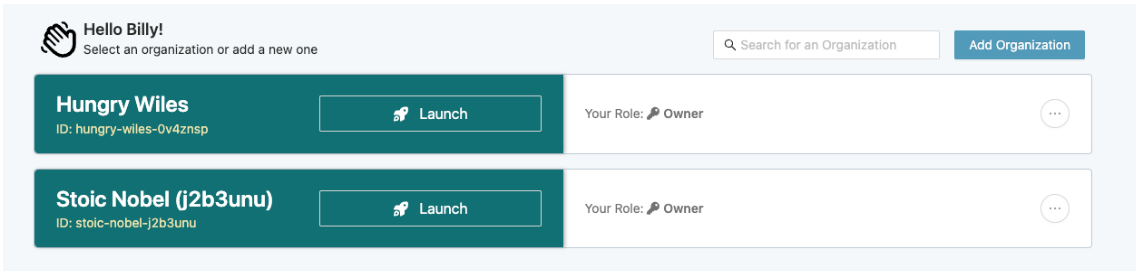
Cribl University's Troubleshooting Criblet on How to Log Into Cribl.Cloud walks you through the whole registration and login flow. To follow the direct course link, first log into your Cribl University account. (To create an account, click the **Sign up** link. You'll need to click through a short **Terms & 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.

2.2. Cribl.Cloud Portal

The Cribl.Cloud portal is the place where you get an overview of and manage your organization, configure network settings, and control user access and billing information.

Select Organization Page

When you own or are a member of multiple Cribl.Cloud Organizations, the Organization selection page – displayed after you first sign in, or sign back in after a logout – enables you to choose which Organization you want to work with. You can later switch your organization via the Account menu.



Select Organization interstitial page

Click any tile's `...` button to reveal an options menu. Here you can check who is the owner of the Organization. You can also click **Leave Organization** 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.)

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.

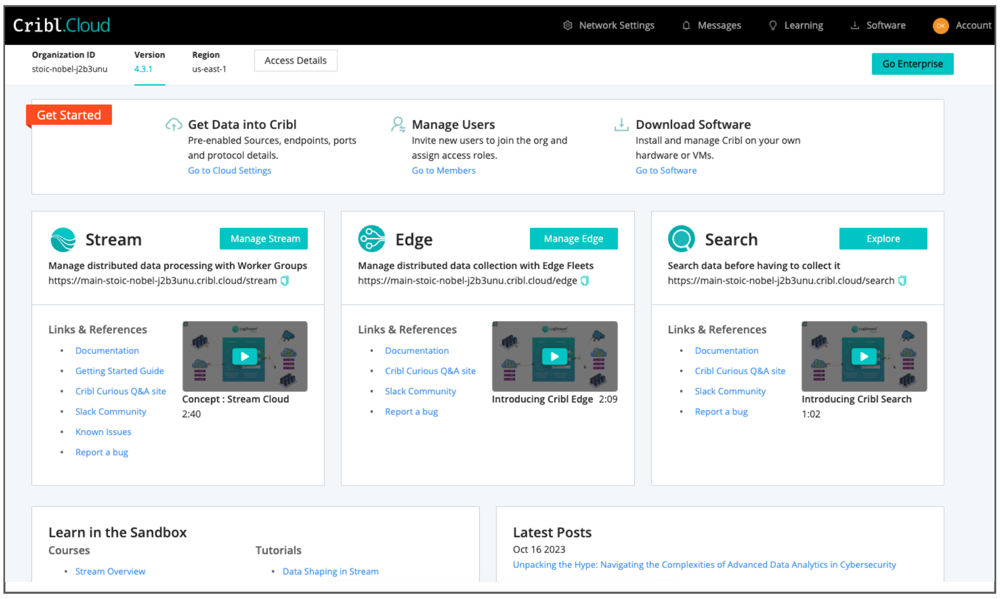
Exploring the Cribl.Cloud Portal

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)
- Network Settings
- Messages
- Learning
- Software
- Account (including Organization details)

Portal Page

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.



Organization ID
stoic-nobel-j2b3unu
Version
4.3.1
Region
us-east-1

Get Data into Cribl
Pre-enabled Sources, endpoints, ports and protocol details.
Go to Cloud Settings

Manage Users
Invite new users to join the org and assign access roles.
Go to Members

Download Software
Install and manage Cribl on your own hardware or VMs.
Go to Software

Stream
Manage distributed data processing with Worker Groups
https://main-stoic-nobel-j2b3unu.cribl.cloud/stream
Links & References
- Documentation
- Getting Started Guide
- Cribl Curious Q&A site
- Slack Community
- Known Issues
- Report a bug

Concept : Stream Cloud 2:40

Edge
Manage distributed data collection with Edge Fleets
https://main-stoic-nobel-j2b3unu.cribl.cloud/edge
Links & References
- Documentation
- Cribl Curious Q&A site
- Slack Community
- Report a bug

Introducing Cribl Edge 2:09

Search
Search data before having to collect it
https://main-stoic-nobel-j2b3unu.cribl.cloud/search
Links & References
- Documentation
- Cribl Curious Q&A site
- Slack Community
- Report a bug

Introducing Cribl Search 1:02

Learn in the Sandbox
Courses
- Stream Overview

Tutorials
- Data Shaping in Stream

Latest Posts
Oct 16 2023
Unpacking the Hype: Navigating the Complexities of Advanced Data Analytics in Cybersecurity

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 Cribl.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 Cribl.Cloud Workers.

Overview and Access Details

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

Organization 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 the following Regions: **us-west-2**, **us-east-1**, **eu-central-1**, and **eu-west-2**.)

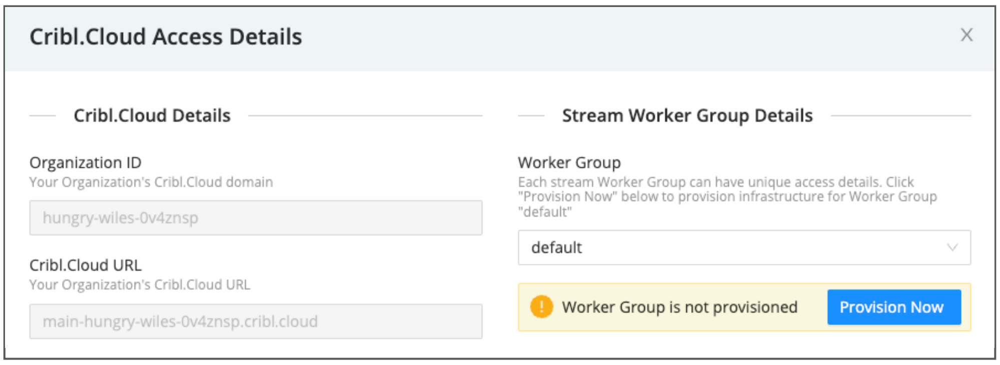
Access Details: See your Organization's details.

The left column repeats the Organization, region, and version information, with one addition:

Cribl.Cloud URL: Static address associated with the load balancer that is in front of the Leader. Hybrid Worker Nodes will connect to this address on port 4200, while the Leader UI is served from this address on port 443.

The 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.


Cribl.Cloud Access Details

Cribl.Cloud Details

Organization ID
Your Organization's Cribl.Cloud domain
hungry-wiles-0v4znsp

Cribl.Cloud URL
Your Organization's Cribl.Cloud URL
main-hungry-wiles-0v4znsp.cribl.cloud

Stream Worker Group Details

Worker Group
Each stream Worker Group can have unique access details. Click "Provision Now" below to provision infrastructure for Worker Group "default"
default

Worker Group is not provisioned

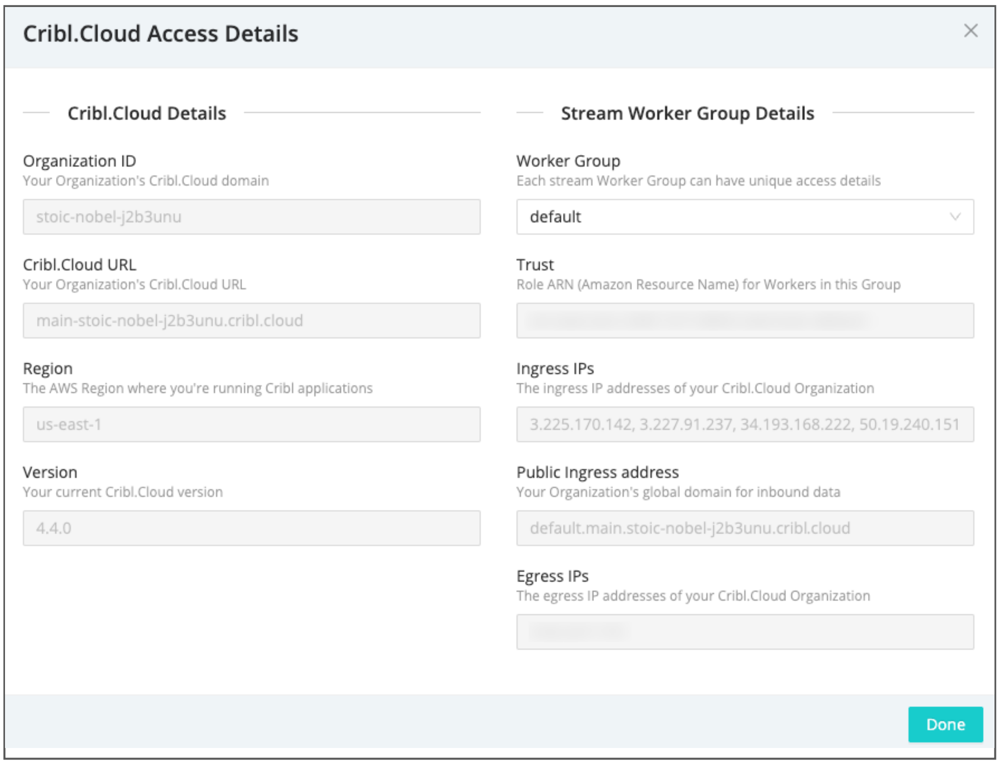
Access Details modal for an unprovisioned Group

- **Trust**: Role ARN for Workers in this Group. You configure these ARNs on the Trust tab.
- **Ingress IPs**: The IPv4 addresses of Worker Nodes' load balancers, also used when receiving data from Push Sources. These addresses will remain constant, so you can build firewall rules around them.

- **Public Ingress address**: Each Group's domain for inbound data. This address prepends the Group name to the Organization's global domain name. It does not append ports per data type – you can obtain these from the Data Sources tab.
- **Egress IPs**: Your Cribl.Cloud Organization's current public IP addresses. These addresses are Group-specific and also dynamic: Cribl will occasionally update them when we need to rescale core infrastructure. The addresses are used for both outbound connections from the Workers and Pull Sources.

Egress IPs are not static.

If you need a static egress IP, contact Cribl Support.



Cribl.Cloud Access Details

Cribl.Cloud Details

Organization ID
Your Organization's Cribl.Cloud domain
stoic-nobel-j2b3unu

Cribl.Cloud URL
Your Organization's Cribl.Cloud URL
main-stoic-nobel-j2b3unu.cribl.cloud

Region
The AWS Region where you're running Cribl applications
us-east-1

Version
Your current Cribl.Cloud version
4.4.0

Stream Worker Group Details

Worker Group
Each stream Worker Group can have unique access details
default

Trust
Role ARN (Amazon Resource Name) for Workers in this Group
[redacted]

Ingress IPs
The ingress IP addresses of your Cribl.Cloud Organization
3.225.170.142, 3.227.91.237, 34.193.168.222, 50.19.240.151

Public Ingress address
Your Organization's global domain for inbound data
default.main.stoic-nobel-j2b3unu.cribl.cloud

Egress IPs
The egress IP addresses of your Cribl.Cloud Organization
[redacted]

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

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

Data Sources

The **Data Sources** tab lists ports, protocols, and data ingestion inputs that are open and available to use, including pre-enabled Sources. Use the **Group** drop-down to filter these details per 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.

For each existing Source listed here, Cribl recommends using the preconfigured endpoint and port to send data into Cribl Stream.

Trust

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 Stream instances.

ACL

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 Stream instances.

Messages

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

Clicking the top nav's **Learning** link opens the **Learning** page, which provides links to everything you need to learn about Cribl Stream 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

If you want to try an on-prem installation, this page offers download links for Cribl Stream, Cribl Edge, and AppScope software. You can download either binary installation files or Docker containers (hosting Ubuntu 20.04), to install and manage on your own hardware or virtual machines.

Account

Manage your Cribl.Cloud account with these submenu items. Hover over **Account** to reveal more options:

- **Profile** allows you to update your personal information.
- **Organization** provides details about the current organization (for an Organization's owner only).
- **Organization Selection** submenu (fly-out) works like the Select Organization page – select a link to traverse to other Organizations.
- **Log Out** signs you out of the account you're on, and takes you back to the login screen.

Organization

Displayed only to an Organization's owner, this page offers information about your Organization's details, Members, and (where applicable) billing and SSO, organized into tabs along the top of the page.

Access to the **Billing** tab includes the **Plan & Invoices** and **Usage** left tabs.

Details Tab

You can add these optional details to make your Cribl.Cloud deployment more recognizable than its randomly generated **Organization ID**:

Field name	Description
Alias	A "friendly" name for your Organization. Upon signing in, members will see this alias above the Organization ID on the Select Organization page.
Description	Add further details about your Organization.
Opt in to beta features	If displayed, this 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 Cribl.Cloud.

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

Members

This tab provides access to inviting and managing other users.

API Management

This tab provides access to existing API credentials and the ability to create new ones.

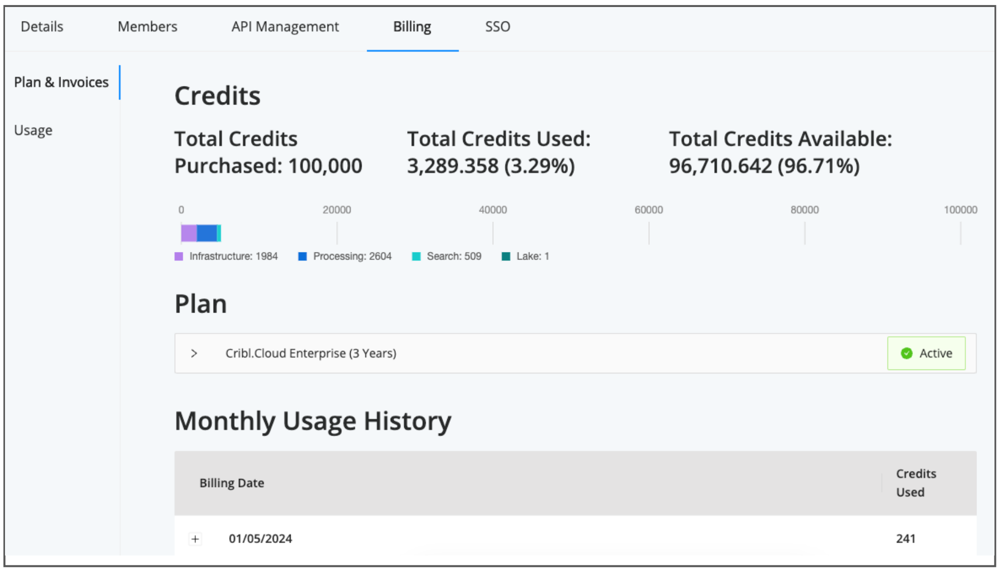
Billing

This tab is displayed only to owners of an Organization on a paid license plan. It provides Plan and Usage left tabs:

Plan & Invoices

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).



Credits

Total Credits Purchased: 100,000 | Total Credits Used: 3,289.358 (3.29%) | Total Credits Available: 96,710.642 (96.71%)

Plan

Cribl.Cloud Enterprise (3 Years) — Active

Monthly Usage History

Billing Date	Credits Used
01/05/2024	241

Billing > Plan & Invoices tab

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

Usage

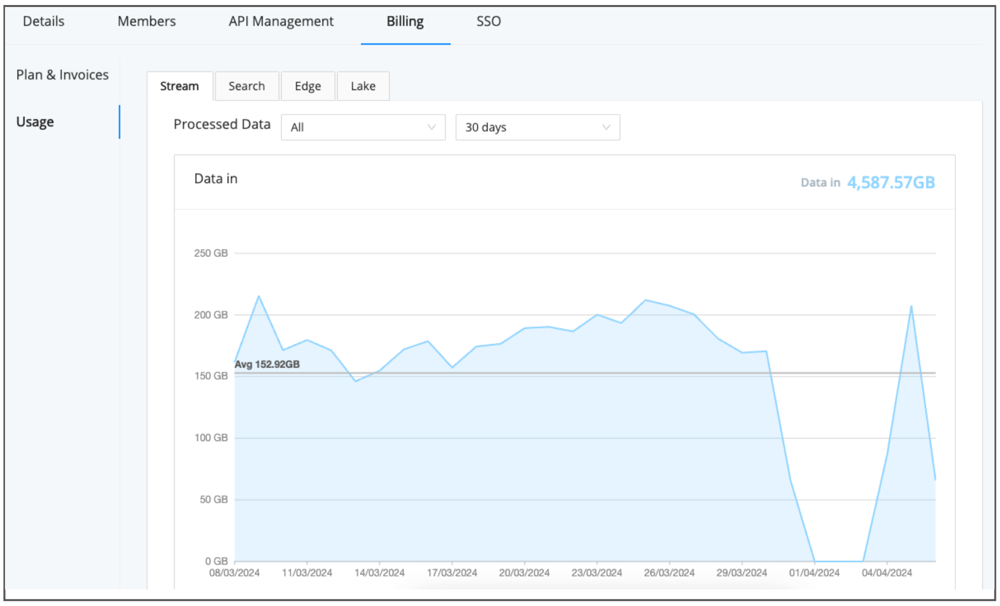
The **Usage** left tab provides nested tabs for **Stream**, **Search**, **Edge**, and **Lake**. (If you're keeping tabs, this is a third level of selectable tabs.)

Select a product to view its graphs, and adjust the duration of time for which data was processed over a selectable trailing period of 7 days to 1 year.

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

The trend line shows daily averages, and you can hover over data points to pop out details.

At the right side of each graph is a total for the selected period.



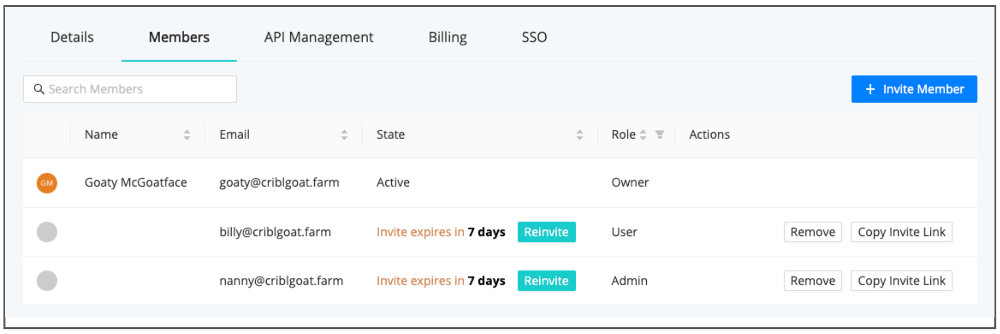
Billing > Usage tab

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.

2.3. Managing Cribl.Cloud

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.

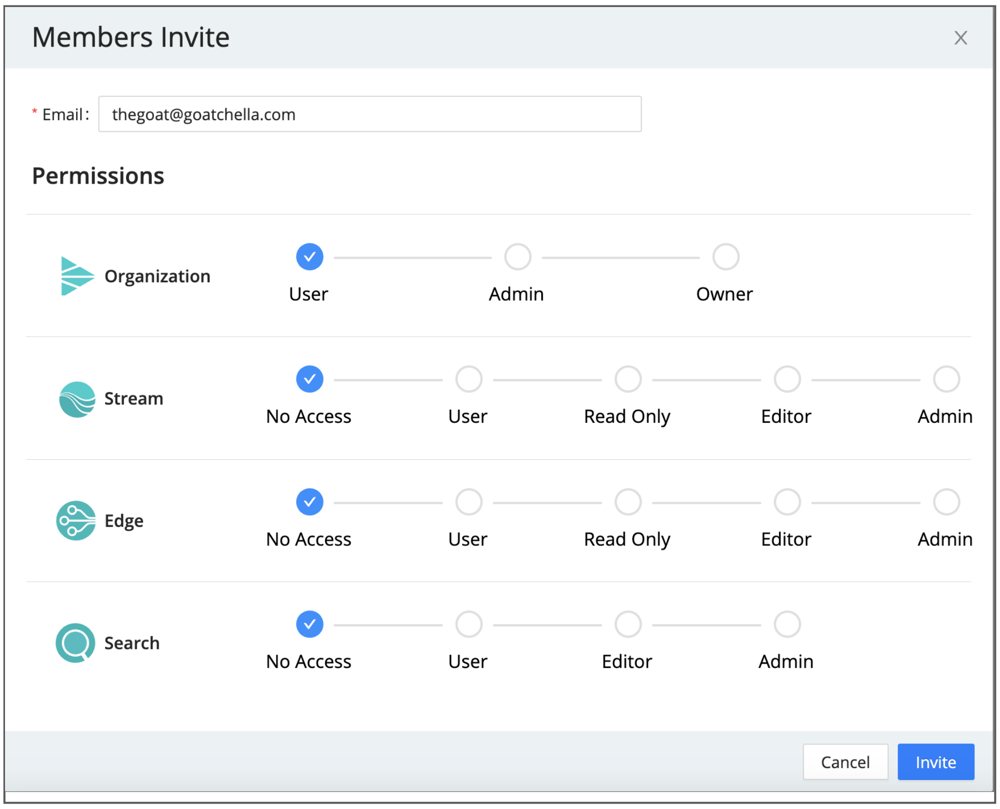


	Name	Email	State	Role	Actions
GM	Goaty McGoatface	goaty@criblgoat.farm	Active	Owner	
		billy@criblgoat.farm	Invite expires in **7 days** Reinvite	User	Remove Copy Invite Link
		nanny@criblgoat.farm	Invite expires in **7 days** Reinvite	Admin	Remove Copy Invite Link

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, and then click **Invite** to send the invitation.



* Email: thegoat@goatchella.com

Permissions

Organization	[✓ User]	Admin	Owner		
Stream	[✓ No Access]	User	Read Only	Editor	Admin
Edge	[✓ No Access]	User	Read Only	Editor	Admin
Search	[✓ No Access]	User	Editor	Admin	

Invite User modal

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 Stream instance at the permission level you've specified.

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 **No Access** permission on each product. An Organization's Owner can assign the **User**, **Admin**, and **Owner** permissions at the Organization level. Assigning the **User** permission requires an Enterprise license.

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

For permissions that you can assign at the lower product, Worker Group, and resource levels (all of those only 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	User	Admin	Owner
Log into the system			
Update own member profile			
View Stream Worker groups you have access to			
`Read Only` Access to all Products			
`Admin` Access to all Products			
View and execute Leader commits			
View and modify Global Settings			
Manage ACLs (Access Control lists)			
View and update SSO settings			
View Data Sources and Trust Policies			

Permission	User	Admin	Owner
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			
View the Organization’s Billing dashboards			
View Organization Details			
Update Organization Details			
Delete an Organization			
View Organization members			
Manage (invite, update, delete) Organization members			

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

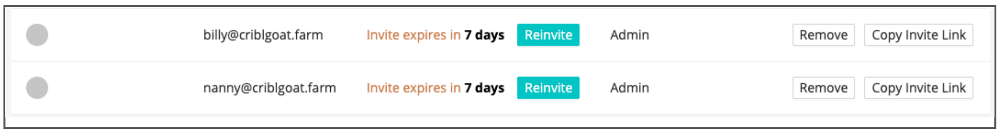
Cribl.Cloud has retired the access-management model of Local Users and Roles/Policies that Cribl applications used prior to v.4.2. Create and configure only **Members** on Cribl applications – references elsewhere in Cribl docs to Local Users do not apply to Cribl.Cloud.

Managing Invites

While an invite is pending, the **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 seven 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.)

2.4. Cribl.Cloud Vs. Self-Hosted

A Cribl.Cloud deployment can differ from an on-prem/customer-managed Cribl Stream deployment. The following documentation lists the main ways in which the Free tier of Cribl.Cloud diverges from a customer-managed deployment. Keep in mind all these differences as you navigate Cribl Stream'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 Stream – 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 Worker Group.

Compared to self-hosted Cribl Stream, 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 Worker Nodes and Worker Groups.

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 Stream uses to encrypt secrets on Worker Groups and Worker 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.

Available only on hybrid, customer-managed Worker Nodes:

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

Available only on self-hosted deployments:

- Top nav's **Settings** > **Global Settings** > **Scripts** (Cribl.Cloud currently does not support configuring or running shell scripts on hybrid or Cribl-managed Worker Nodes.)

Available on Cribl-managed Edge Nodes, but unavailable on Cribl-managed Workers:

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

Persistent Queues can be configured on both hybrid and Cribl-managed Worker Nodes, with an Enterprise plan. On hybrid Worker Nodes, you can freely define the **Max queue size**, based on the disk space you provision. On Cribl-managed Worker 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.)

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 Sources and ports already enabled for you, plus 11 additional TCP ports (`20000-20010`) that you can use to add and configure more Cribl Stream Sources.

TLS Details

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

Cribl HTTP and Cribl TCP Sources/Destinations

Use the Cribl HTTP Destination and Source, and/or the Cribl TCP Destination and Source, to relay data between Worker 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.

2.5. Enterprise Cloud

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

- Configuring and managing multiple Worker Groups and Fleets.
- Notifications within Cribl apps, to PagerDuty, and/or to other services via webhook.
- Fine-grained, role-based control of Member Permissions on your Cribl.Cloud Organization and on individual product resources.
- Single sign-on/SSO authentication from external identity providers.
- The hybrid deployment option, described just below.

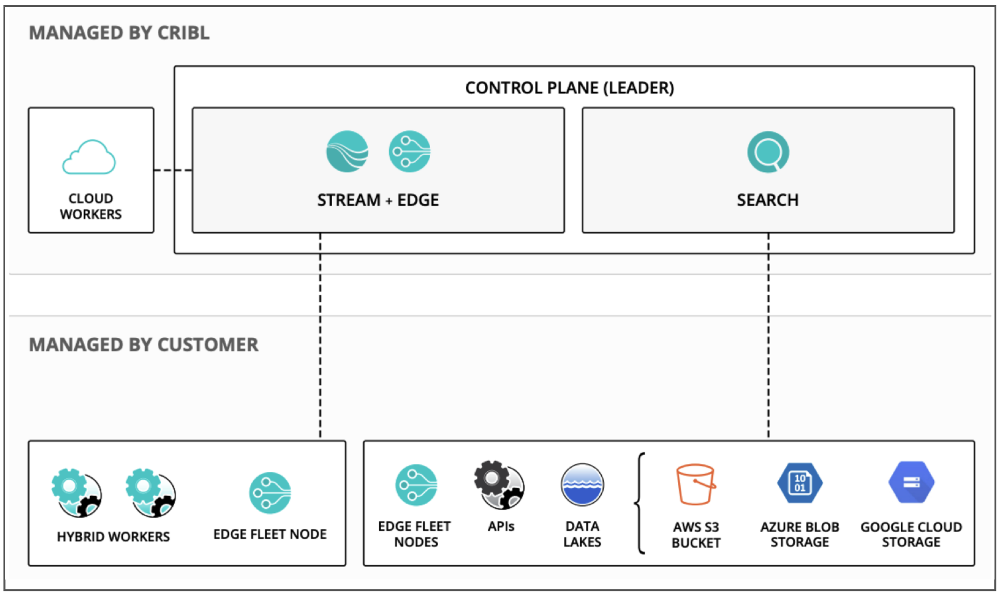
With an Enterprise Plan, the Leader resides in Cribl.Cloud, and controls a flexible mix of Cribl-managed and/or customer-managed Worker Groups. Cribl manages the Leader's high availability on your behalf.

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

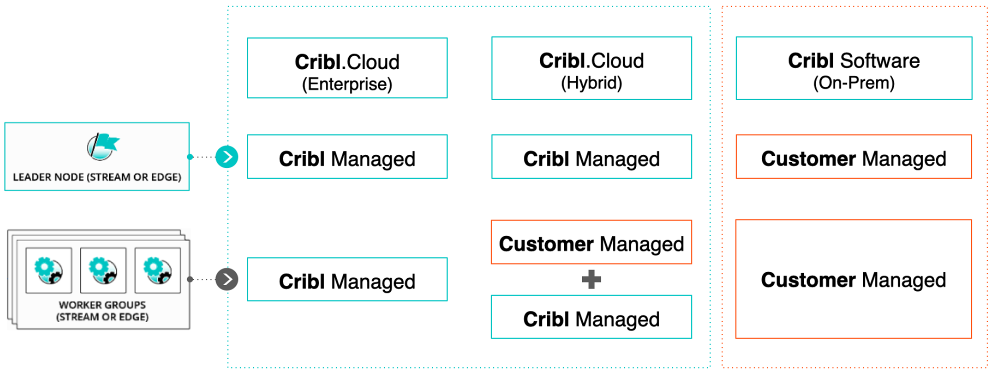
Hybrid Deployment

The diagrams below show the comparative flexibility of a hybrid Cribl.Cloud deployment. The Leader (control plane) resides in Cribl.Cloud, while the Workers that process the data can be in any combination of the following environments:

- In Cribl.Cloud, managed by Cribl.
- In public or private cloud instances that you manage.
- On-premises in your data centers.



Enterprise hybrid deployment, with control plane and Cribl-managed Workers in Cribl.Cloud



Enterprise hybrid deployment, with only control plane in Cribl.Cloud

As the footprint of your operations grows or changes, this flexibility makes it easy to reconfigure Cribl Stream in tandem. You can rapidly expand Cribl Stream 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 Worker Groups, by easily auto-generating Stream or Edge command-line scripts within Cribl Stream'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 Worker Group than the Cribl-managed `default` Group – which can contain its own Worker Nodes.

- All Worker Nodes' hosts must allow outbound communication to the Cribl.Cloud Leader's port 4200 at `https://main-<Organization-name>.cribl.cloud:4200`, to enable configuration and workload management by the Leader.
- On all Worker Nodes' hosts, firewalls must allow outbound communication on port 443 to the Leader and to `https://cdn.cribl.io`. This port is also used to bootstrap hybrid Workers from the Leader.
- All Worker 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 Worker Node instances that you manage.

Adding (Bootstrapping) Workers

To add Workers to your hybrid Cribl.Cloud deployment, Cribl recommends that you use the script outlined in Bootstrap Workers from Leader. Hosts for the new Workers must open the same ports (4200 and 443) listed in Hybrid Requirements.

You have three options for generating the script, outlined in these subsections of the Bootstrap topic linked above:

- Auto-generate it from the Leader's UI.
- Make a `GET` API request to the Leader.
- curl the same API request.

In Cribl Edge, you access all these bootstrap options via the Manage Edge Nodes page's **Add/Update Edge Node** control.

Hybrid Cribl HTTP/TCP Configuration

If you use the Cribl HTTP Destination and Source pair, or the Cribl TCP Destination and Source pair, to relay data between Worker Nodes connected to the same Leader, configuring hybrid Workers demands particular care.

The Worker Nodes that host each pair's Destination and Source must specify exactly the same Leader Address. Otherwise, token verification will fail – breaking the connection, and preventing data flow. In hybrid Cribl.Cloud deployments, the Leader's Address format is `main-<your-Org-ID>.cribl.cloud`. When configuring a hybrid Worker, use that format in the **Address** field.

To configure hybrid Workers:

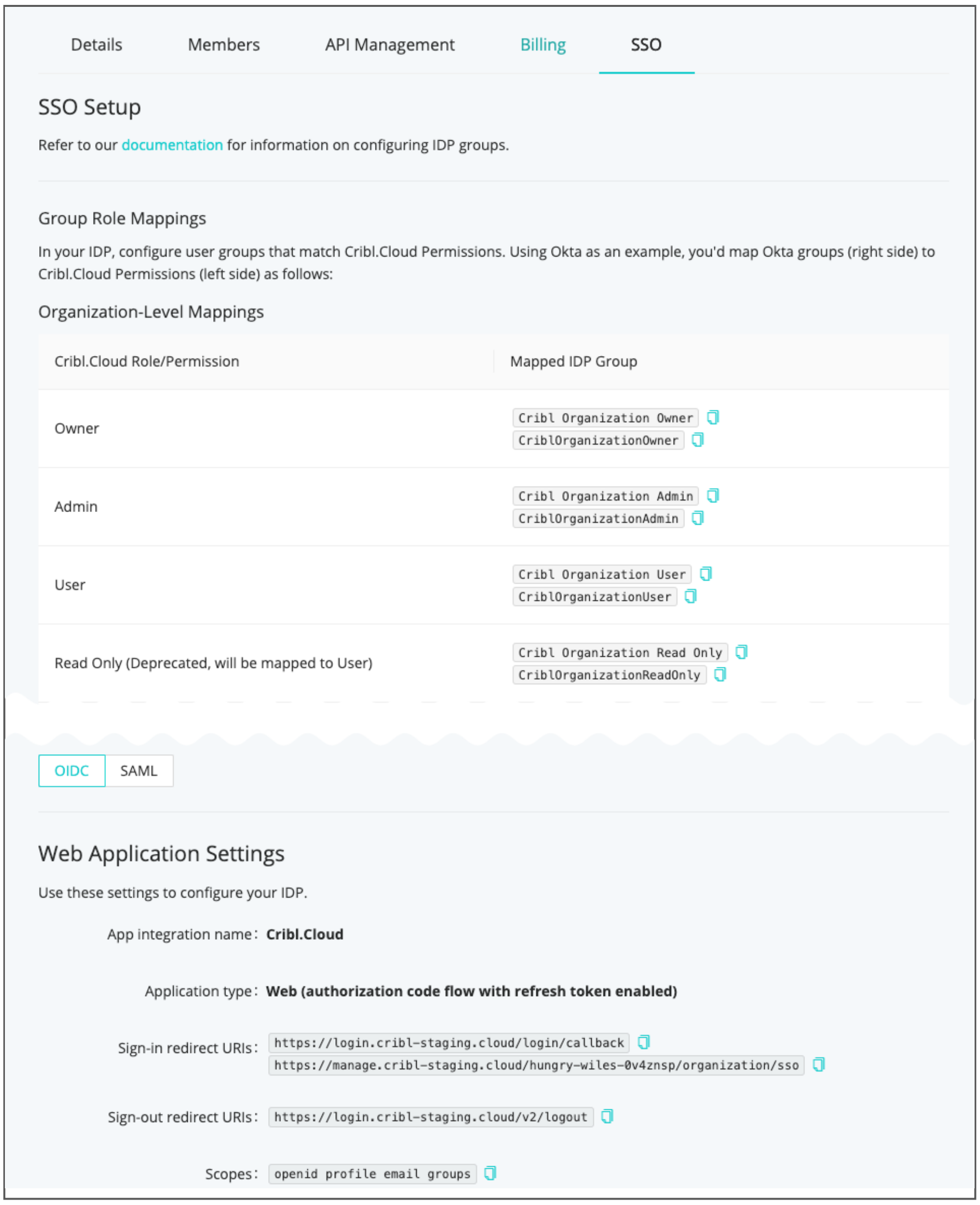
1. Log directly into their UI, then select **Settings** > **Global Settings** > **Distributed Settings**. Make sure the **Mode** is set to **Managed Worker** or **Managed Edge** (which might require a restart).
2. Then select the **Leader Settings** left tab, and ensure a consistent entry in the **Address**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

2.6. 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/Microsoft Entra ID 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 Stream 4.0, you can submit these details directly on your Cribl.Cloud portal's Organization page.


Details Members API Management Billing SSO

SSO Setup

Refer to our documentation for information on configuring IDP groups.

Group Role Mappings

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

Organization-Level Mappings

Cribl.Cloud Role/Permission	Mapped IDP Group

Owner	`Cribl Organization Owner`
	`CriblOrganizationOwner`

Admin	`Cribl Organization Admin`
	`CriblOrganizationAdmin`

User	`Cribl Organization User`
	`CriblOrganizationUser`

Read Only (Deprecated, will be mapped to User)	`Cribl Organization Read Only`
	`CriblOrganizationReadOnly`

OIDC SAML

Web Application Settings

Use these settings to configure your IDP.

App integration name:	**Cribl.Cloud**
Application type:	**Web (authorization code flow with refresh token enabled)**
Sign-in redirect URIs:	`https://login.cribl-staging.cloud/login/callback` `https://manage.cribl-staging.cloud/hungry-wiles-0v4znsp/organization/sso`
Sign-out redirect URIs:	`https://login.cribl-staging.cloud/v2/logout`
Scopes:	`openid profile email groups`

Organization page SSO tab

This section covers both sides of the process. For additional details specifically about integrating Cribl Stream 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.

2.6.1. Common SSO Setup Steps

This page lists initial preparation steps that are the same for all Single Sign-On (SSO) configurations, whether you're using OIDC or SAML.

Set Up Fallback Access

In your Cribl.Cloud Organization, ensure that at least one Owner creates a local account, using an email domain that's separate from the corporate domain on which you're configuring SSO. This will ensure backup access if SSO configuration breaks.

Avoiding Being Forced into SSO

By design, SSO does Home Realm Discovery (HRD), meaning that when you enter your email to log in, SSO checks the domain specified in the email address; and, if the domain matches a connected SSO provider, SSO sends you to that provider to log in.

Normally, this is a good thing. One exception is when you want to sign up for additional Cribl.Cloud orgs without being forced through the SSO for an existing Cribl.Cloud org. In that case, try the following workaround:

1. Edit the login URL to delete the word `identifier`.
2. Use the edited URL to log in; instead of forcing you through SSO, Cribl.Cloud will ask for a username and password.

For example:

- Original URL: `https://login.cribl.cloud/u/login/identifier?state=<long_string_of_characters>`
- Edited URL: `https://login.cribl.cloud/u/login/?state=<long_string_of_characters>`

Configure Groups

In your IDP (identity provider), configure user groups that match Permissions for both Cribl.Cloud Organizations and individual products.

Using Okta as an example, you'd map Okta groups (right side) to Cribl.Cloud Roles (left side) as follows.

IDP Group Naming

The names you define for groups in your IDP must include `Cribl` and the Role name (`Owner`, `Admin`, or `User`). They can include `Organization` or product name (`Stream`, `Edge`, or `Search`). If they don't contain a product name, the group is treated as an Organization name.

Even though IDP group names without Organization will be treated as Organization-level permissions, we recommend keeping Organization in the name for clarity.

You can use either the open or the closed format (with or without spaces) in group names. You can freely add prefixes or suffixes to Group Names that follow the formats in the examples above. Cribl will ignore these additions when mapping IDP groups to Cribl Roles. Examples:

- `SOME-LABEL-12345-CriblOrganizationOwner`
- `CriblOrganizationEditor-420`

Cribl.Cloud Role/Permission	IDP Group Name
Owner	Cribl Organization Owner /or/ CriblOrganizationOwner
Admin	Cribl Organization Admin /or/ CriblOrganizationAdmin
User	Cribl Organization User /or/ CriblOrganizationUser
Editor	**(Deprecated, will be mapped to Admin)** Cribl Organization Editor /or/ CriblOrganizationEditor
Read Only	**(Deprecated, will be mapped to User)** Cribl Organization Read Only /or/ CriblOrganizationReadOnly

Considerations for Microsoft (Azure) AD

For the groups claim configuration, you must select `Groups assigned to the application` for association. The source attribute must be set to `sAMAccountName`. Enable **Emit group name for cloud-only groups** to also return the `sAMAccountName` attribute for cloud-only groups. See the Microsoft Entra ID + OpenID Configuration topic.

Default Product Permissions

When you map external users to your Cribl Organization, their initial product-level Permissions follow a different inheritance pattern than Members configured within Cribl. This is to avoid downgrading product-level Permissions that Organization-level `User`s might already have.

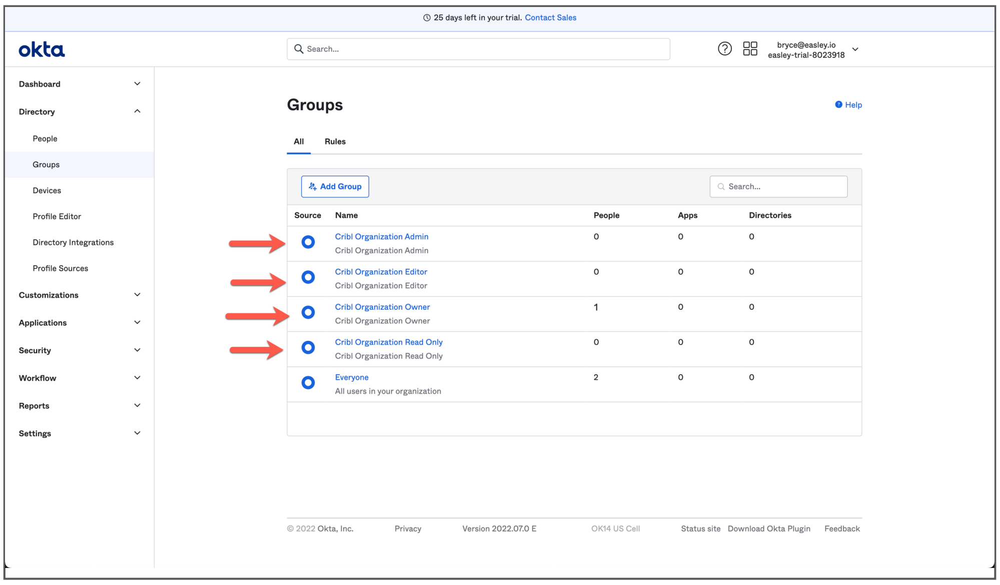
The defaults for mapped users are:

- Organization `Owner` or `Admin` inherits `Admin` Permission on all products.
- Organization `User` inherits `User` Permission on all products.
- Organization `Editor` (deprecated) inherited `editor` legacy Roles on all products.
- Organization `Read Only` (deprecated) inherited `Read Only` Permission on Stream and Edge, and `User` Permission on Search.

Group Configuration Better Practices

- A Cribl.Cloud Organization's Owner Role can be shared and transferred among multiple users. This facilitates gradual ownership transfers, corporate reorganizations, and other scenarios.
- Those users should be in both the Owner and Admin groups in your IDP. (This enables them to acquire all needed permissions across Cribl's two corresponding Roles.)
- Aside from dual-assigning the Owners, you should assign every other user only one group in your IDP. (Cribl's Admin and Editor Roles include all the permissions of the Roles below them.)

Here's an example of how groups configuration (at an early stage) might look in Okta's UI:



Source	Name	People	Apps	Directories
○	Cribl Organization Admin Cribl Organization Admin	0	0	0
○	Cribl Organization Editor Cribl Organization Editor	0	0	0
○	Cribl Organization Owner Cribl Organization Owner	1	0	0
○	Cribl Organization Read Only Cribl Organization Read Only	0	0	0
○	Everyone All users in your organization	2	0	0

Mapping Okta groups to Cribl.Cloud Roles

2.6.2. 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, navigate to the **Applications** section of your Okta environment and click **Create App Integration**.

Next, configure the app integration with the options below.

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

General Settings

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

- **App integration name**: `Cribl.Cloud (OIDC)`
- **Grant type**: Select `Authorization Code` and `Refresh Token`.
- **Refresh token behavior**: Select `Use persistent token`.
- **Sign-in redirect URIs**:
  - https://login.cribl.cloud/login/callback
  - `https://manage.cribl.cloud/<organizationID>/organization/sso`
- **Sign-out redirect URIs**: https://login.cribl.cloud/v2/logout

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

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

Assignments

Configure the **Assignments** pane with the following options:

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

Sign-On Tab

In the **OpenID Connect ID Token** section, set the **Groups claim filter** to: `groups` : `Starts with` : `Cribl`.

To obtain the **Issuer URL** you'll need to provide to Cribl in the next section, change the value in the **Issuer** field from `Dynamic` to `Okta URL`.

This step concludes the setup procedure for Okta (or other IDP).

Submit Your App Info to Cribl

Next, provide Cribl 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.

The **Web Application Settings** are prefilled for you, so you only need to fill in the **Cribl Cloud SSO Settings** section with the following details from your IDP client configuration:

- Client ID
- Client Secret
- Issuer URL. Copy this value from the **API** > **Settings** section of your Okta environment.

OIDC/Okta Chiclet Setup (Optional)

If you want to initiate login from your Okta instance with OIDC authentication configured, an Okta admin can configure an app integration as follows:

1. From Okta's left nav, select the **Applications** page.
2. Find the OIDC application created earlier in the OIDC/Okta Setup Example.
3. Click that application, and select **General Settings** > **Edit**.
4. In the **Initiate login URI** field, enter `https://manage.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).
5. 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.

2.6.3. 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	Microsoft Entra ID
Single Sign-On URL	Reply URL (Assertion Consumer Service URL)
Audience URI	Identifier (Entity ID)

Create an Enterprise Application

In Microsoft Entra ID:

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

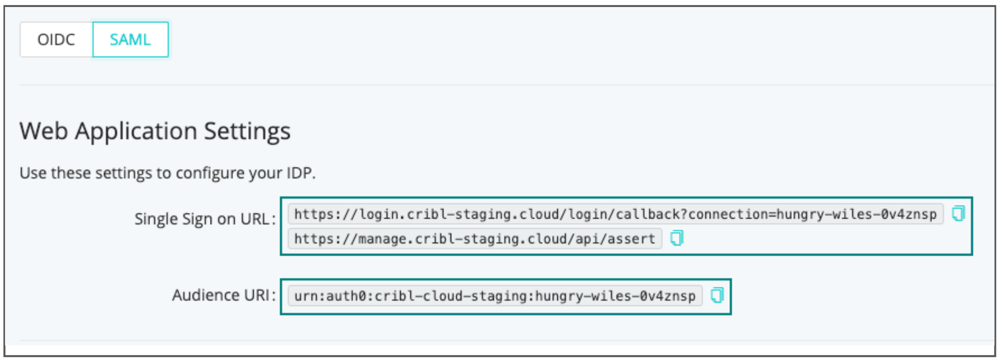
From Microsoft Entra ID's left nav:

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 Microsoft Entra ID'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**.



Web Application Settings

Use these settings to configure your IDP.

Single Sign on URL: `https://login.cribl-staging.cloud/login/callback?connection=hungry-wiles-0v4znsp` `https://manage.cribl-staging.cloud/api/assert`

Audience URI: `urn:auth0:cribl-cloud-staging:hungry-wiles-0v4znsp`

SSO information for configuring Azure AD

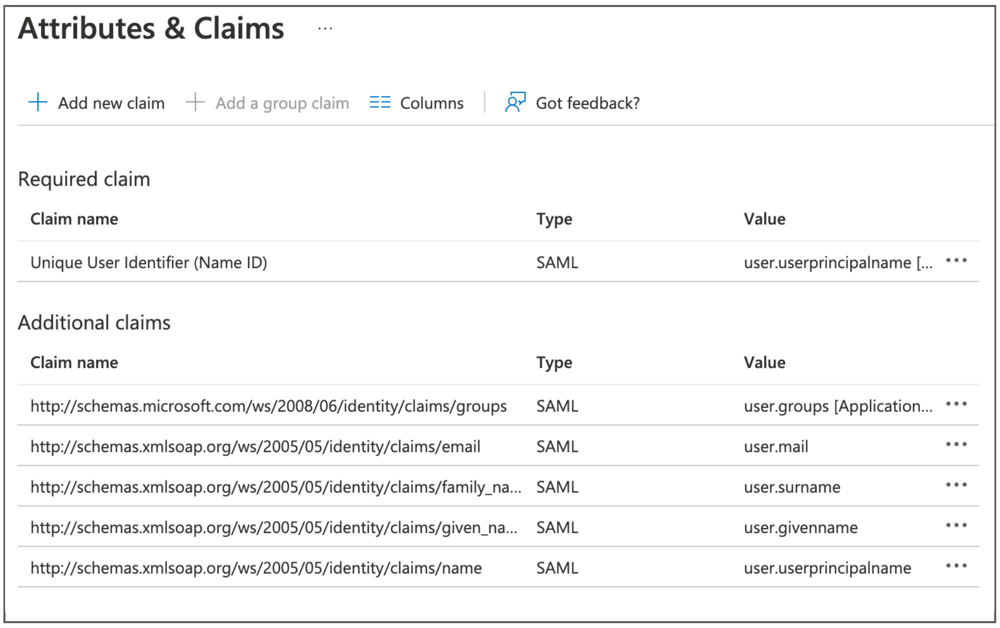
Configure Attributes and Groups Claims

In Microsoft Entra ID, 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.



Attributes & Claims

Required claim

Claim name	Type	Value
Unique User Identifier (Name ID)	SAML	user.userprincipalname [...

Additional claims

Claim name	Type	Value
http://schemas.microsoft.com/ws/2008/06/identity/claims/groups	SAML	user.groups [Application...
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/email	SAML	user.mail
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/family_na...	SAML	user.surname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/given_na...	SAML	user.givenname
http://schemas.xmlsoap.org/ws/2005/05/identity/claims/name	SAML	user.userprincipalname

SAML Microsoft Entra ID 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.

SAML/Azure AD Setup with My Apps Chiclet (Optional)

If you want to log into Cribl.Cloud via the Microsoft My Apps chiclet, complete the following procedure:

1. In Microsoft Entra ID, navigate to the enterprise application that you created to integrate SSO.
2. From the left nav, select **Single Sign-on**.
3. In the Enterprise Application's **Basic SAML Configurations** UI, click **Edit**.
4. In the **Sign on URL (Optional)** section, enter the following URL:

```
https://portal.cribl.cloud/login?connection=<organizationID>
```

You also need to allow self-service access to the Cribl App, or assign AD groups permissions to access the application.

Link Existing Users

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

2.6.4. 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=<$organizationID>`
2. Test SSO URL.
   Required to test your connection. Okta accepts this under "Other Requestable SSO URLs," and this is the second URL that your **SSO** > **SAML** tab lists under "Single sign-on URL." Example:
   `https://manage.cribl.cloud/api/assert`
3. Entity ID.
   Okta calls this an "Audience URI (SP Entity ID)," and your **SSO** > **SAML** tab calls it just an "Audience URI." Example: `urn:auth0:cribl-cloud-prod:<$organizationID>`

Create SAML 2.0 App Integration

1. To create your app integration within Okta, navigate to the **Applications** section of your Okta environment and click **Create App Integration**.
2. Next, create the app integration with **Sign-in method**: `SAML 2.0`.

Configure SAML Settings

1. 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`.

2. 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://manage.cribl.cloud/api/assert`
- **Index**: Set this to `0`.

Configure Attribute Statements

Configure **Attribute Statements** for these attributes, as shown below:

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

Then save your app integration.

Submit Your App Info to Cribl

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

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

2. Return to your Okta environment and click **View SAML setup instructions**.

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://manage.cribl.cloud/login?connection=<organizationID>` (where `<organizationID>` is your Cribl.Cloud Organization's ID).
7. Click **Done**.
8. Click **Assign** and assign all of the Cribl.Cloud groups to the application.
9. The Cribl.Cloud chiclet should now be available for all users in the Cribl groups you've assigned.

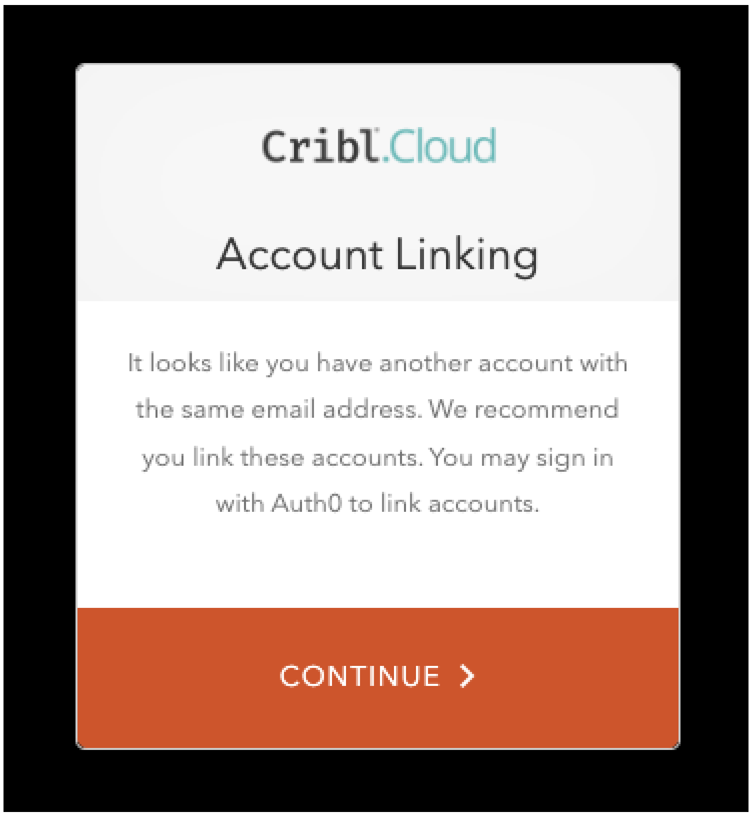
Link Existing Users

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

2.6.5. 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.)



Cribl.Cloud

Account Linking

It looks like you have another account with the same email address. We recommend you link these accounts. You may sign in with Auth0 to link accounts.

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. CrowdStream

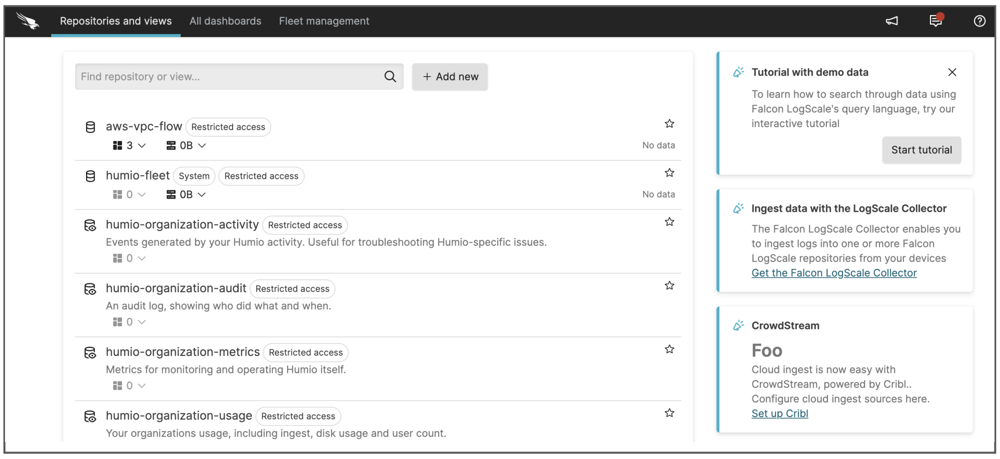
CrowdStream is a special Cloud-hosted version of Cribl Stream, available through CrowdStrike Falcon LogScale. This collaboration is tailored to forward data from a wide range of sources into LogScale for XDR analysis and log management.

En route, you can process events using all of Stream's options for aggregating, sampling, redacting, cloning, and otherwise shaping and directing your data. CrowdStream provides parallel options to route data to cost-effective Amazon S3 storage, and to S3 data lakes, for later replay and analysis.

Getting Started

These steps assume that you've set up your CrowdStrike Falcon LogScale account, obtained your corresponding API token, and logged into LogScale.

1. On the **CrowdStream** tile at right, click the **Set up Cribl** link.



Events generated by your Humio activity. Useful for troubleshooting Humio-specific issues.

An audit log, showing who did what and when.

Metrics for monitoring and operating Humio itself.

Your organizations usage, including ingest, disk usage and user count.

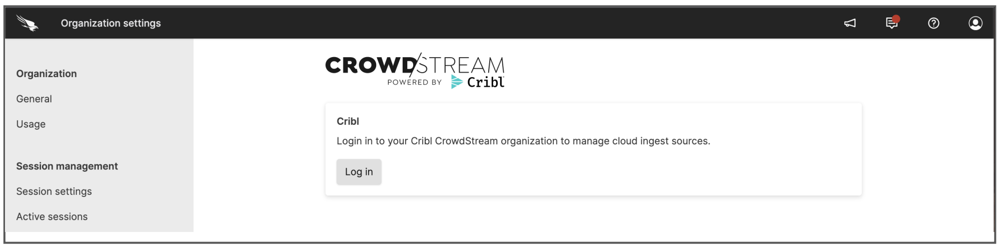
To learn how to search through data using Falcon LogScale's query language, try our interactive tutorial

The Falcon LogScale Collector enables you to ingest logs into one or more Falcon LogScale repositories from your devices

Cloud ingest is now easy with CrowdStream, powered by Cribl.. Configure cloud ingest sources here.

Navigating from LogScale to CrowdStream

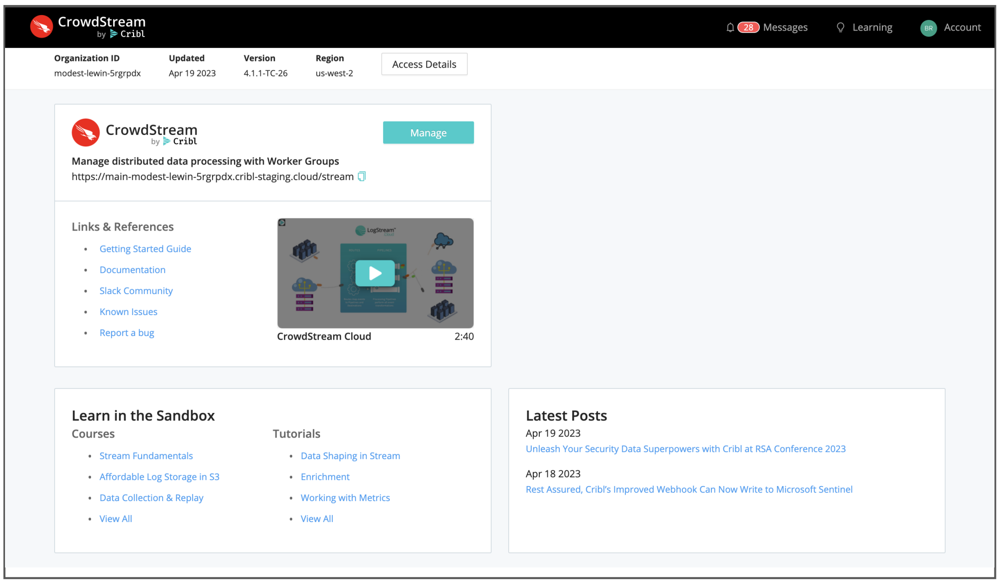
2. On the resulting **Organization settings** page, click **Log in**.



Login in to your Cribl CrowdStream organization to manage cloud ingest sources.

Logging into CrowdStream

3. This brings you to the CrowdStream home page. Click **Manage** on the **CrowdStream** tile at left.



Organization ID: modest-lewin-5rgrpdx
Updated: Apr 19 2023
Version: 4.1.1-TC-26
Region: us-west-2

Manage distributed data processing with Worker Groups
https://main-modest-lewin-5rgrpdx.cribl-staging.cloud/stream

Links & References
- Getting Started Guide
- Documentation
- Slack Community
- Known Issues
- Report a bug

Learn in the Sandbox

Courses
- Stream Fundamentals
- Affordable Log Storage in S3
- Data Collection & Replay
- View All

Tutorials
- Data Shaping in Stream
- Enrichment
- Working with Metrics
- View All

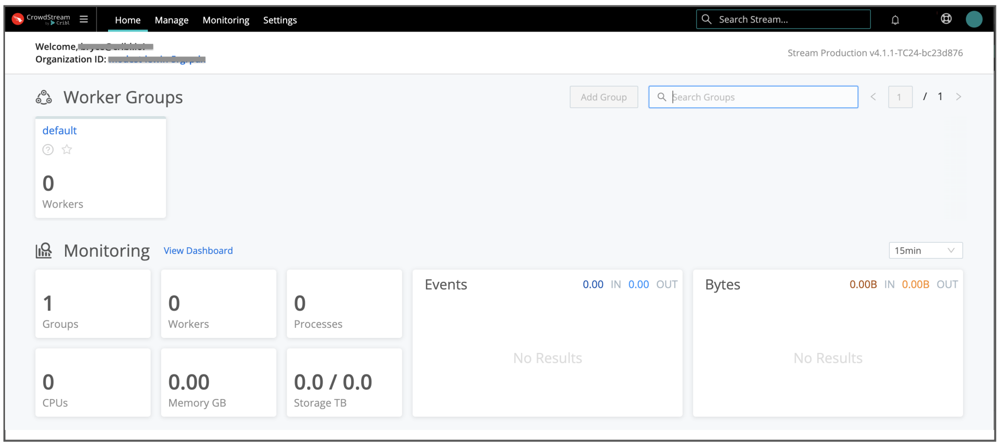
Latest Posts

Apr 19 2023
Unleash Your Security Data Superpowers with Cribl at RSA Conference 2023

Apr 18 2023
Rest Assured, Cribl's Improved Webhook Can Now Write to Microsoft Sentinel

Selecting the CrowdStream app

4. On CrowdStream's **Worker Groups** page, click the **default** Group's tile at the upper left.
   (A Worker Group is a collection of processing instances that share the same configuration. For details, see Cribl Stream's Distributed Deployment topic.)



Welcome, [redacted]
Organization ID: [redacted]
Stream Production v4.1.1-TC24-bc23d876

Worker Groups

default
0 Workers

Monitoring

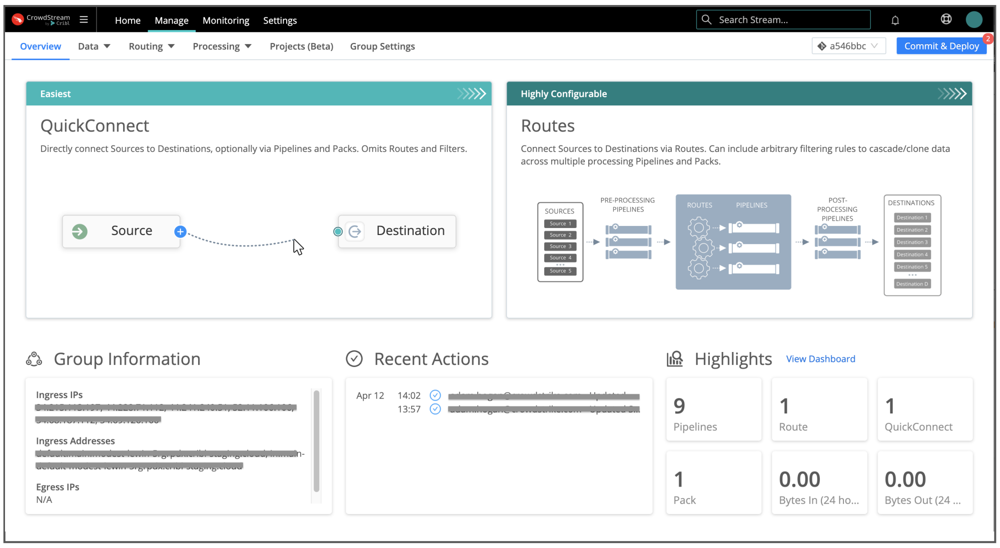
1 Groups
0 Workers
0 Processes
0 CPUs
0.00 Memory GB
0.0 / 0.0 Storage TB

Events 0.00 IN 0.00 OUT
No Results

Bytes 0.00B IN 0.00B OUT
No Results

Selecting the **default** Group

5. This brings you to the **Manage** page of CrowdStream's top nav, and to the **Overview** tab of this Group's submenu.
   To begin configuring this Group, you now have a two-up choice of **QuickConnect** or **Routes** tiles. How to choose? See the next section.



QuickConnect

Directly connect Sources to Destinations, optionally via Pipelines and Packs. Omits Routes and Filters.

Routes

Connect Sources to Destinations via Routes. Can include arbitrary filtering rules to cascade/clone data across multiple processing Pipelines and Packs.

Group Information

Ingress IPs
[redacted]

Ingress Addresses
[redacted]

Egress IPs
N/A

Recent Actions

Apr 12 14:02 [redacted]
13:57 [redacted]

Highlights

9 Pipelines	1 Route	1 QuickConnect
1 Pack	0.00 Bytes In (24 ho...	0.00 Bytes Out (24 ...

Group **Overview** page

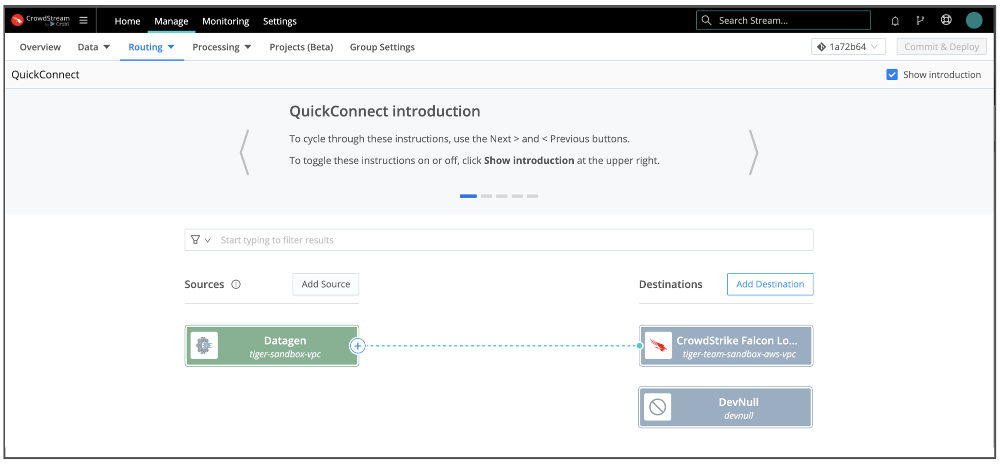
For More Information

- Cribl.Cloud Launch Guide

Simple or Complex Routing?

Your first choice is how simply or intricately you want to configure data flow from the first data Source you set up. CrowdStream offers two user interfaces:

- QuickConnect (the left tile shown above) provides a simple visual UI. You drag and drop connections from Sources on the left to Destinations on the right. Each connection can include as much or as little intermediate processing as you like. To remove a connection, just drag it off the Destination (you'll get a confirmation modal). The only practical limitation here is that data flow through every Source -> Destination connection must be parallel and independent.
- Data Routing (the right tile shown above) provides fine-grained control, via a more traditional UI that you configure across separate modals and pages. You build a Routing table with conditional processing. This can cascade and clone data across multiple processing Routs, according to the arbitrary filtering rules and sequence that you define.



QuickConnect introduction

To cycle through these instructions, use the Next > and < Previous buttons.

To toggle these instructions on or off, click **Show introduction** at the upper right.

Configuring data flow in QuickConnect

So the trade-off between these two options is ease of use versus flexibility. These two options are separate, but not mutually exclusive: You can set up simultaneous data flows through QuickConnect and Data Routes.

You can also move Sources, and their connections to Destinations, between the two UIs. So one obvious option is to configure your first data flow in QuickConnect, and then move to the Routing UI as your familiarity and needs expand.

Here is a lightning-fast (one-minute) preview of **all** the configuration steps below, in QuickConnect:

CrowdStream QuickStart

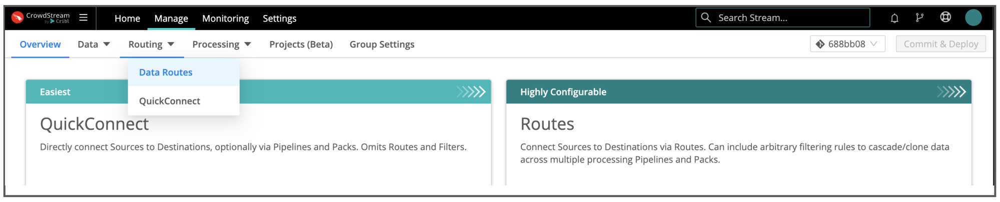
01:14

Navigate Between User Interfaces

To work in QuickConnect, click the left tile shown above, or select **Routing** > **QuickConnect**.

To work in the Routing UI, click the right tile shown above, or select **Routing** > **Data Routes**.

You can always switch between the two environments from a Group's **Manage** > **Overview** tab, by using the **Routing** submenu.



QuickConnect

Directly connect Sources to Destinations, optionally via Pipelines and Packs. Omits Routes and Filters.

Routes

Connect Sources to Destinations via Routes. Can include arbitrary filtering rules to cascade/clone data across multiple processing Pipelines and Packs.

Traversing between QuickConnect and Data Routes

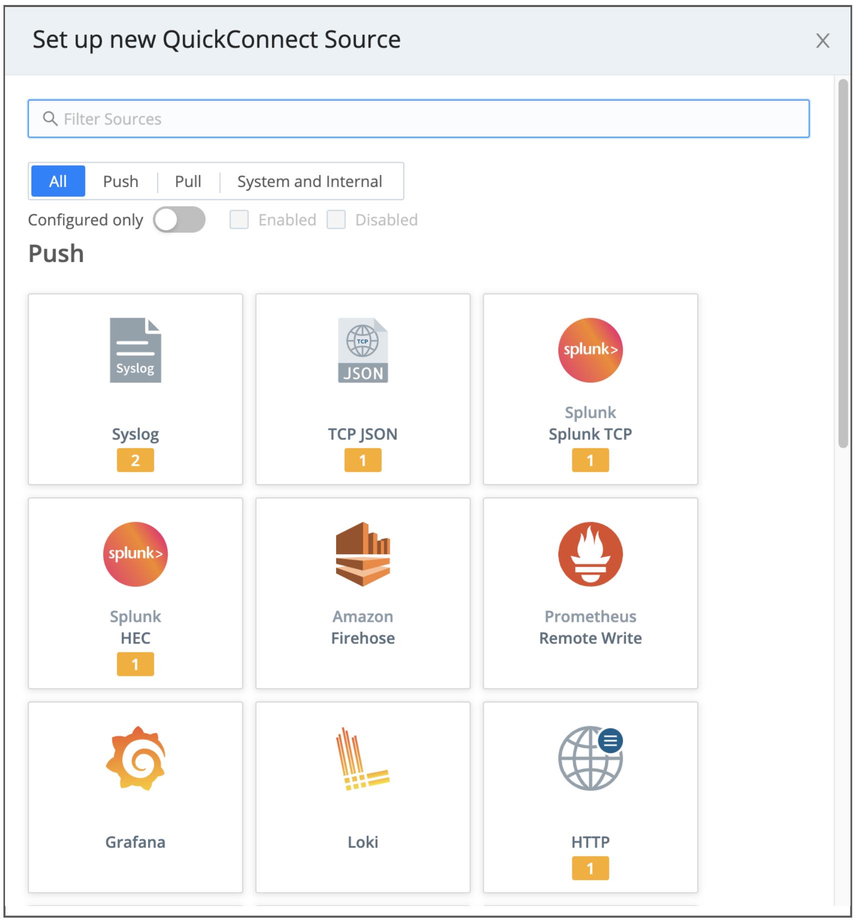
Configure a Source

A CrowdStream Source is a model of an upstream service sending data into the system. In this model, you configure authentication and/or other needed parameters.

You can configure multiple instances of a given Source type, each with different parameters. You can also attach processing that's specific to each instance. CrowdStream provides the Datagen Source type to generate sample inbound events for development and testing.

Using QuickConnect

In QuickConnect, click the **Add Source** button on the left (Sources) side. In the resulting left drawer, search for and click your desired Source type to begin configuring a new instance of that Source.

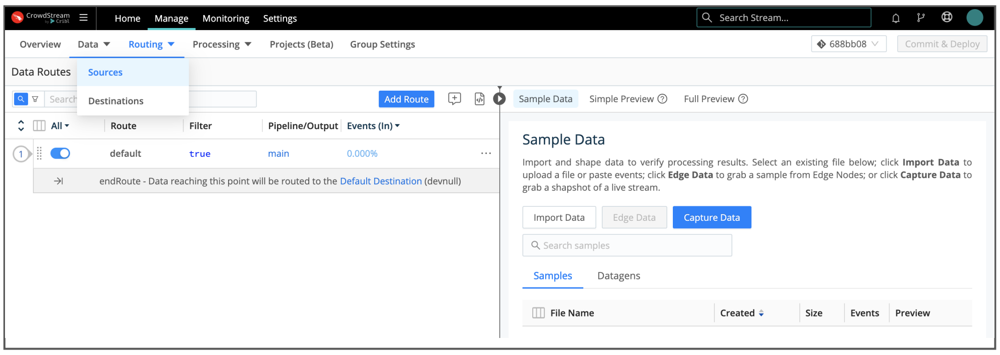


Selecting a Source through the QuickConnect UI

To configure (or reconfigure) a Source that's already present in QuickConnect, hover over its tile and click the **Configure** button. Note that for multiple Sources of the same type, tiles will display stacked until you hover over the stack.

Using Data Routes

In the Routing UI, first click **Data** > **Sources**.



Route	Filter	Pipeline/Output	Events (In)
default	true	main	0.000%

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Sample Data

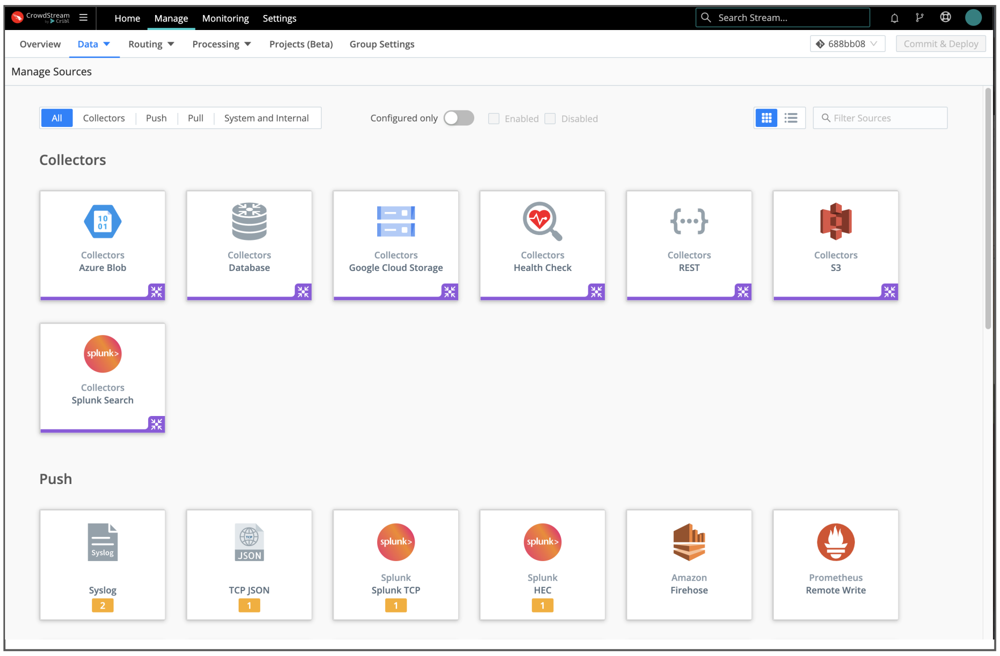
Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

File Name	Created	Size	Events	Preview

Selecting a Source through the Routing UI

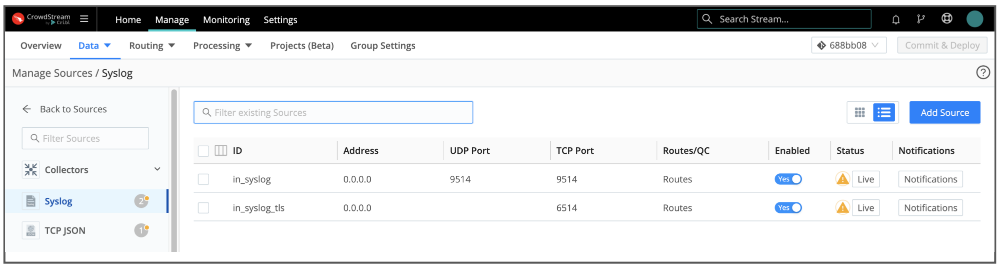
On the resulting **Manage Sources** page, search for your desired Source type, then click its tile to begin configuring a new instance of that Source.

(After you click through to a Source, the same set of peer Sources will be available in the left nav.)



Manage Sources page

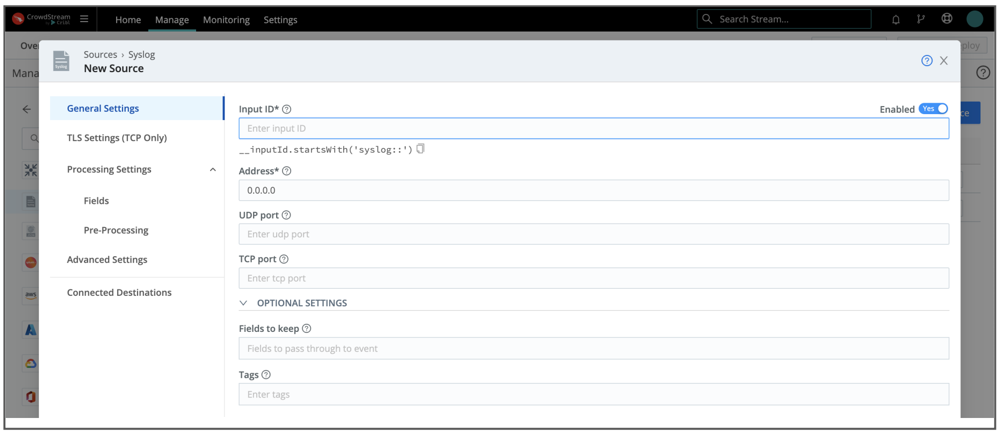
Click an existing Source's row here to access its configuration, or click **Add Source** to configure a new instance of that Source.



ID	Address	UDP Port	TCP Port	Routes/QC	Enabled	Status	Notifications
in_syslog	0.0.0.0	9514	9514	Routes	Yes	Live	Notifications
in_syslog_tls	0.0.0.0		6514	Routes	Yes	Live	Notifications

Manage Syslog Sources UI

You then configure (or reconfigure) the Source in a modal like this:



Input ID* — Enter input ID — Enabled: Yes
`__inputId.startsWith('syslog::')`
Address*: 0.0.0.0
UDP port — Enter udp port
TCP port — Enter tcp port
OPTIONAL SETTINGS
Fields to keep — Fields to pass through to event
Tags — Enter tags

Syslog config modal

For More Information

- Sources
- Using Datagens
- QuickConnect
- Routes

Configure a Destination

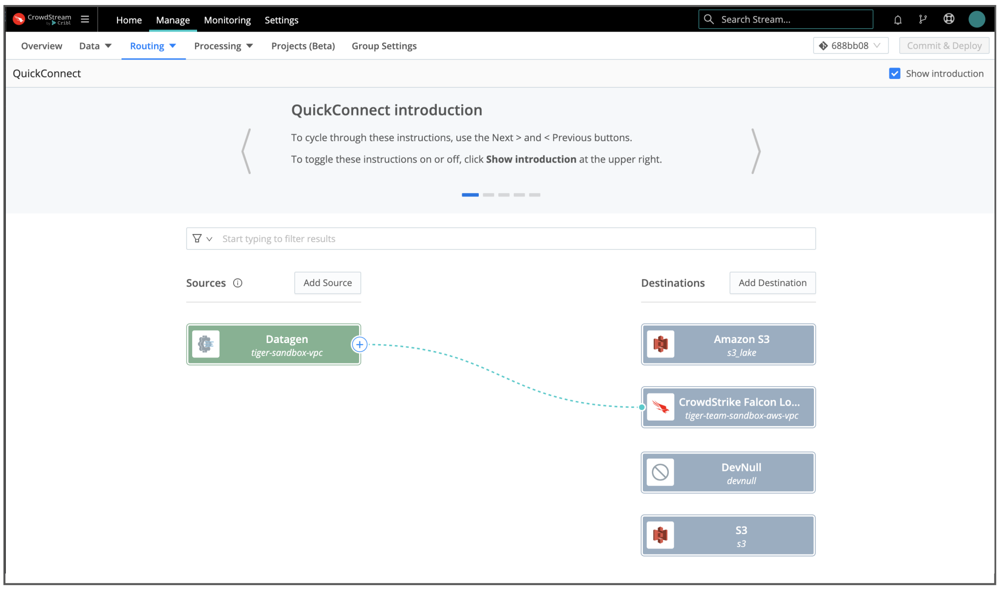
A CrowdStream Destination is a model of a downstream receiver of events, enabling you to configure authentication and/or other needed parameters.

As with Sources, you can configure multiple instances of a given Destination type, each with different parameters. You can attach processing that's specific to each instance. CrowdStream provides the following Destinations:

- CrowdStrike Falcon LogScale – Set up your LogScale endpoint, token, and other parameters here.
- Amazon S3 Compatible Stores – Route full-fidelity data to cost-effective storage for later replay and analysis.
- Data Lakes > Amazon S3 – Store full-fidelity data for later retrieval or analysis. Compatible with Cribl Search (a separate product).
- DevNull – This output simply consumes and discards events, to facilitate development and testing.

Using QuickConnect

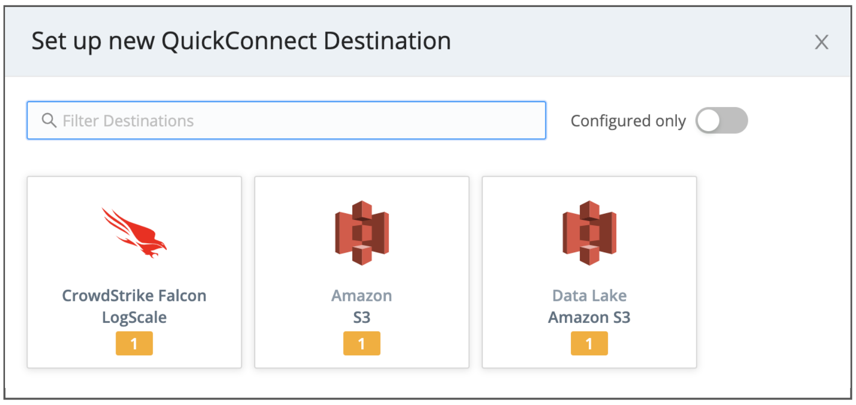
To connect a configured Source to a Destination tile already present on the right, just drag from the Source and drop the connector onto the Destination. A pop-up will prompt you to configure the connection via a Passthru, Pipeline or Pack (explained below).



Multiple Destinations added to QuickConnect

To configure (or reconfigure) the Destination, hover over its tile and click the **Configure** button. For multiple Destinations of the same type, tiles will display stacked until you hover over the stack.

To add a new Destination, click the **Add Destination** button on the right (Destinations) side. In the resulting right drawer, search for and click your desired Destination type to begin configuring a new instance of that Destination.

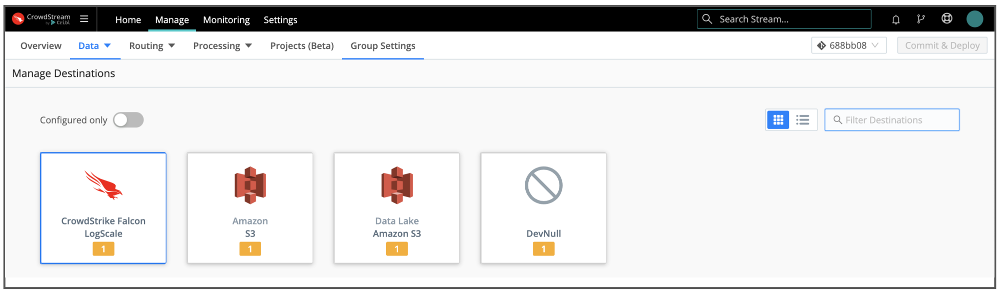


Selecting additional Destinations in QuickConnect

Using Data Routes

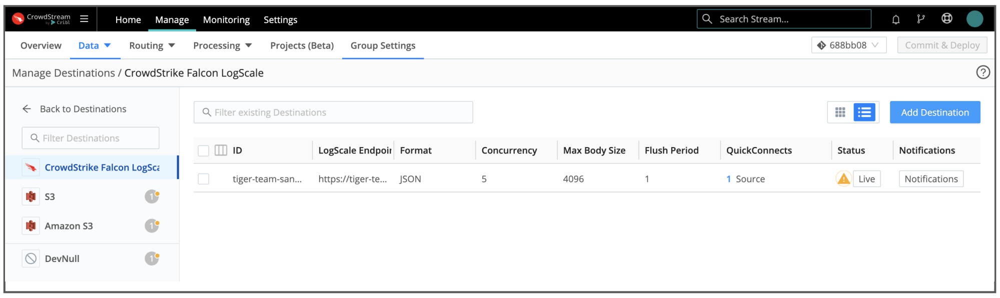
In the Routing UI, first click **Data** > **Destinations**. On the resulting **Manage Destinations** page, search for your desired Destination type, then click its tile to begin configuring it.

(After you click through to a Destination, the same set of peer Destinations will be available in the left nav.)



Manage Destinations page

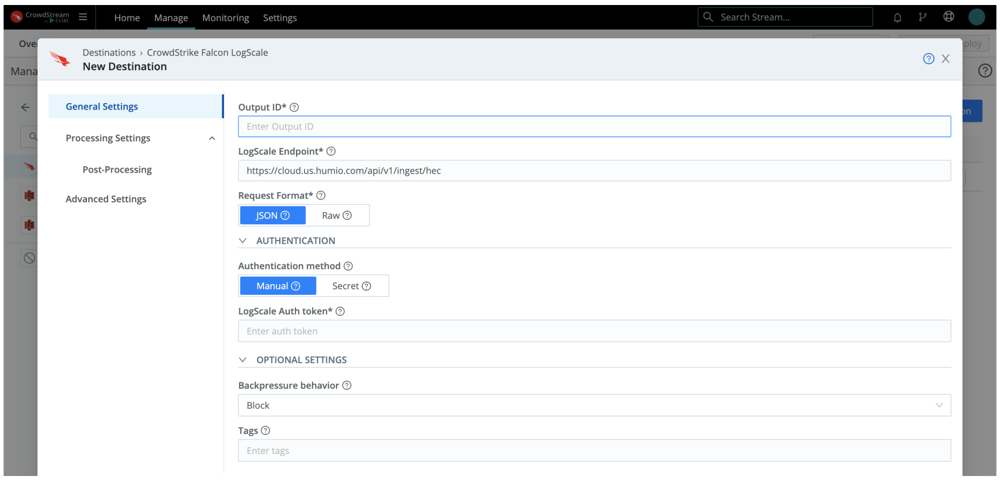
Click an existing Destination's row here to access its configuration, or click **Add Destination** to configure a new instance of that Destination.



ID	LogScale Endpoint	Format	Concurrency	Max Body Size	Flush Period	QuickConnects	Status	Notifications
tiger-team-san...	https://tiger-te...	JSON	5	4096	1	1 Source	Live	Notifications

Manage LogScale Destinations UI

You then configure (or reconfigure) the Destination in a modal like this:



Output ID*: Enter Output ID
LogScale Endpoint*: https://cloud.us.humio.com/api/v1/ingest/hec
Request Format*: [✓ JSON] Raw
Authentication method: [✓ Manual] Secret
LogScale Auth token*: Enter auth token
Backpressure behavior: Block
Tags: Enter tags

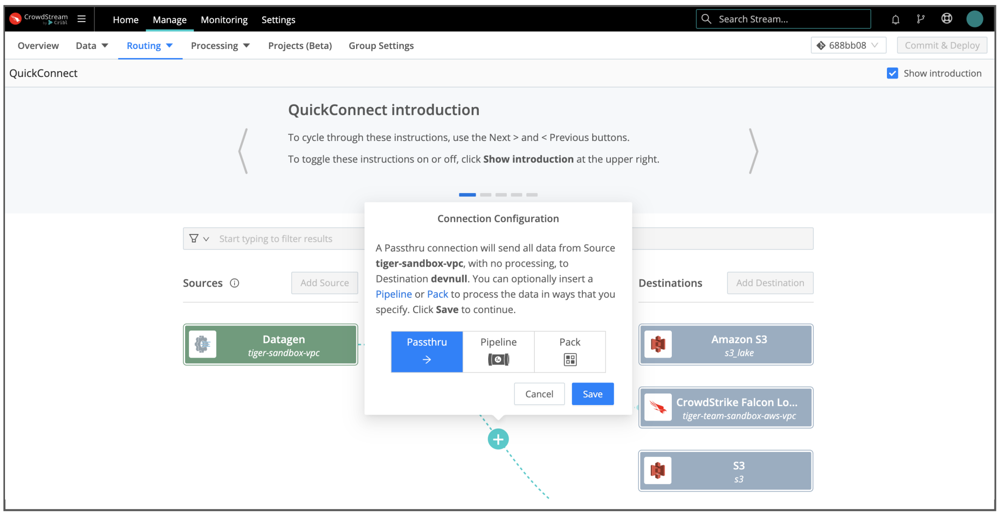
LogScale config modal

For More Information

- See the individual Destination topics linked at the head of this section.

Connect: Passthru, Pipeline, or Pack

In QuickConnect, as soon as you drag and drop a Source -> Destination connection, a pop-up dialog will prompt you to select a connection type: **Passthru**, **Pipeline**, or **Pack**. (In Data Routes, you'll make the same choices elsewhere.)



QuickConnect introduction

To cycle through these instructions, use the Next > and < Previous buttons.

To toggle these instructions on or off, click **Show introduction** at the upper right.

Connection Configuration

A Passthru connection will send all data from Source **tiger-sandbox-vpc**, with no processing, to Destination **devnull**. You can optionally insert a Pipeline or Pack to process the data in ways that you specify. Click **Save** to continue.

Connection-type selector in QuickConnect

What are these options?

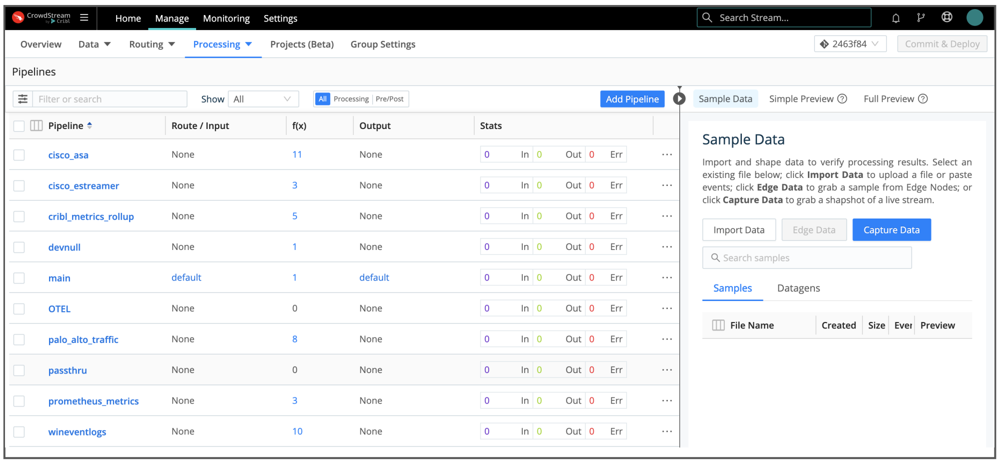
- A **Passthru** establishes a simple direct connection between a Source -> Destination pair.
- A Pipeline hosts a stack of Functions to process your data, in the order you arrange them.
- A Pack is a microcosm of a whole CrowdStream configuration, designed to share configs between Worker Groups or deployments. A Pack can encapsulate its own Pipelines, Routes, Knowledge libraries, and sample data (but not data Sources or Destinations). A Pack can snap in wherever you can use a Pipeline.

So let's unpack Pipelines, where you'll define most of your event processing.

Pipeline Options

When you first create a new Pipeline, it's empty, making it equivalent to **Passthru**. CrowdStream provides a set of out-of-the-box Pipelines for specific purposes. As you can see below, one of these is a **passthru** Pipeline. This similarly contains no Functions – making it the Data Routes UI's equivalent of QuickConnect's **Passthru** button.

For these reasons, it's easy to start with an empty Pipeline, and then add Functions as you need them.



Pipeline	Route / Input	f(x)	Output	Stats
cisco_asa	None	11	None	0 In 0 Out 0 Err
cisco_estreamer	None	3	None	0 In 0 Out 0 Err
cribl_metrics_rollup	None	5	None	0 In 0 Out 0 Err
devnull	None	1	None	0 In 0 Out 0 Err
main	default	1	default	0 In 0 Out 0 Err
OTEL	None	0	None	0 In 0 Out 0 Err
palo_alto_traffic	None	8	None	0 In 0 Out 0 Err
passthru	None	0	None	0 In 0 Out 0 Err
prometheus_metrics	None	3	None	0 In 0 Out 0 Err
wineventlogs	None	10	None	0 In 0 Out 0 Err

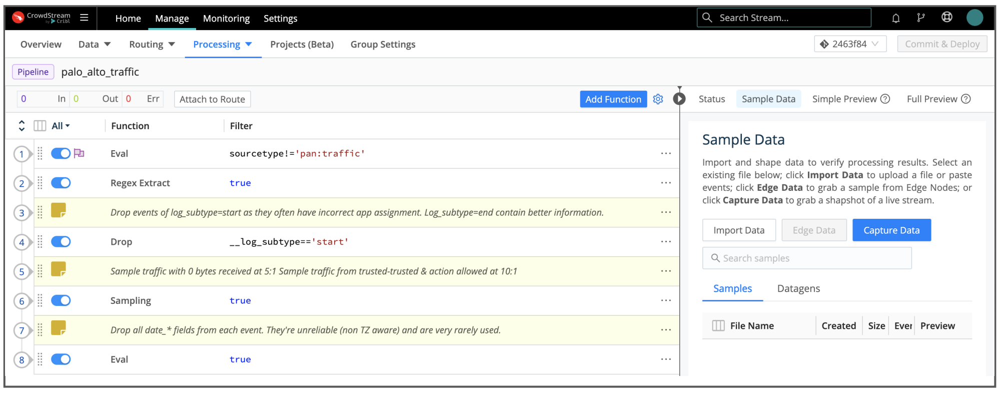
Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

Pipelines Shipped with CrowdStream

A Pipeline, as you populate it, is a stack of processing Functions and optional comments. Use the accordions on the left to expand or collapse each Function. Use the toggles on the left to enable/disable each Function – preserving structure as you develop and debug.

Use the right **Sample Data** pane to preview how the Pipeline's current state transforms a set of events. (Controls here enable you to capture live data, save it to sample or Datagen files, and reload those files.)



Pipeline palo_alto_traffic

#	Function	Filter
1	Eval	`sourcetype!='pan:traffic'`
2	Regex Extract	`true`
3	*Drop events of log_subtype=start as they often have incorrect app assignment. Log_subtype=end contain better information.*	
4	Drop	`__log_subtype=='start'`
5	*Sample traffic with 0 bytes received at 5:1 Sample traffic from trusted-trusted & action allowed at 10:1*	
6	Sampling	`true`
7	*Drop all date_* fields from each event. They're unreliable (non TZ aware) and are very rarely used.*	
8	Eval	`true`

Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

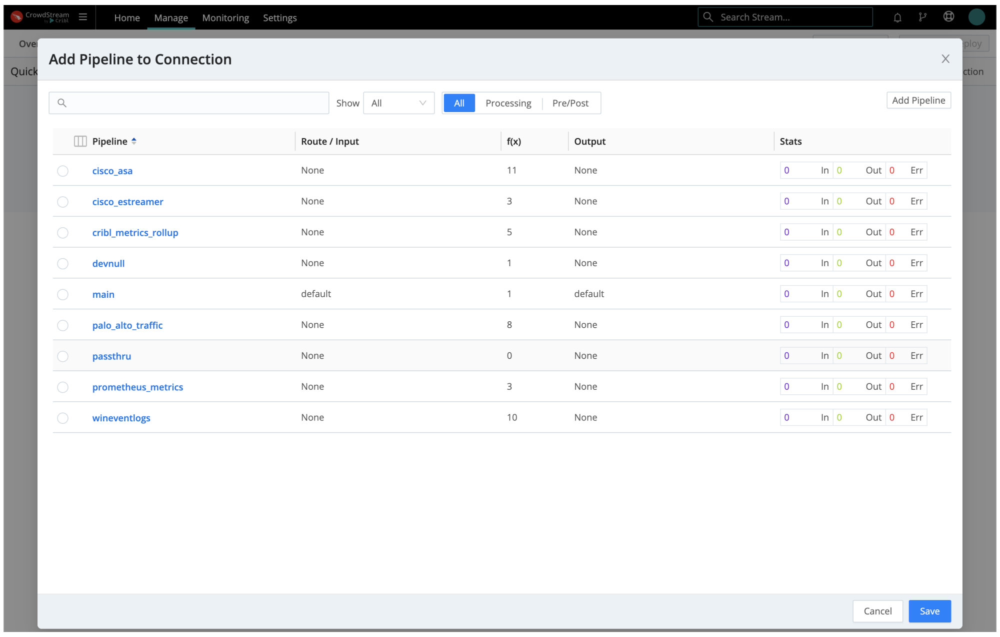
A Pipeline's Functions stack

Although you'll configure most data shaping in processing Pipelines between Sources and Destinations, you can also attach a pre-processing Pipeline to any Source instance, and a post-processing Pipeline to any Destination instance. Use these options when you need to "condition" the specific data type handled by a particular inbound or outbound integration.

Connect via QuickConnect

To select a connection type in QuickConnect, click the appropriate button in the pop-up dialog shown above. To accept the default **Passthru** option, just click **Save**. Your Source and Destination will now be connected.

If you select **Pipeline**, you'll see an **Add Pipeline to Connection** modal. Here, you can click to select an existing Pipeline, or click **Add Pipeline** to define a new one. Then click **Save** here to establish the connection.



Pipeline	Route / Input	f(x)	Output	Stats
cisco_asa	None	11	None	0 In 0 Out 0 Err
cisco_estreamer	None	3	None	0 In 0 Out 0 Err
cribl_metrics_rollup	None	5	None	0 In 0 Out 0 Err
devnull	None	1	None	0 In 0 Out 0 Err
main	default	1	default	0 In 0 Out 0 Err
palo_alto_traffic	None	8	None	0 In 0 Out 0 Err
passthru	None	0	None	0 In 0 Out 0 Err
prometheus_metrics	None	3	None	0 In 0 Out 0 Err
wineventlogs	None	10	None	0 In 0 Out 0 Err

Add Pipeline to Connection modal

Selecting **Pack** opens an **Add Pack to Connection** modal that works exactly the same way.

Whichever option you choose here, your connection between this Source and Destination is now complete.

To get data flowing, jump ahead to commit and deploy your new configuration.

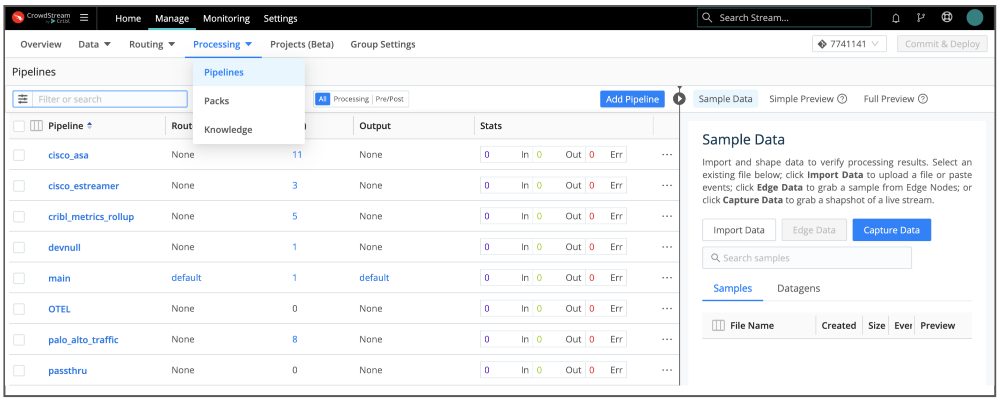
Connect via Data Routes

The Data Routes UI enables you to finely configure multiple, interdependent connections between Sources and Destinations. Events clone and cascade across a Routing table, based on the filtering rules and sequence you define.

In exchange for this flexibility, you sacrifice QuickConnect's one-stop, snap-together UI metaphor. After configuring your Sources and Destinations – which you've already done above – you need to connect them by separately setting up at least one Pipeline and one Route.

Pipelines Access

From a Group's **Manage** tab, select **Processing** > **Pipelines**. In addition to the minimal **passthru** Pipeline identified above, other simple building blocks here include **main** (which initially contains only a simple Eval Function) and **devnull** (which contains a single Function that just drops events).



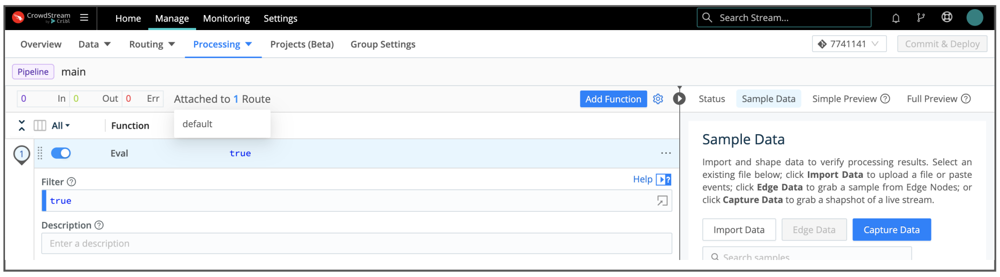
Pipeline	Rout		Output	Stats
cisco_asa	None	11	None	0 In 0 Out 0 Err
cisco_estreamer	None	3	None	0 In 0 Out 0 Err
cribl_metrics_rollup	None	5	None	0 In 0 Out 0 Err
devnull	None	1	None	0 In 0 Out 0 Err
main	default	1	default	0 In 0 Out 0 Err
OTEL	None	0	None	0 In 0 Out 0 Err
palo_alto_traffic	None	8	None	0 In 0 Out 0 Err
passthru	None	0	None	0 In 0 Out 0 Err

Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

Routing to Pipelines

Out-of-the-box Pipelines are already connected to Routes. When you click into a Pipeline, you can hover to display the connected Route, as shown here. But as our final connection step, let's next look at how to connect up everything (Source, Pipeline, Destination) in a Route.



Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

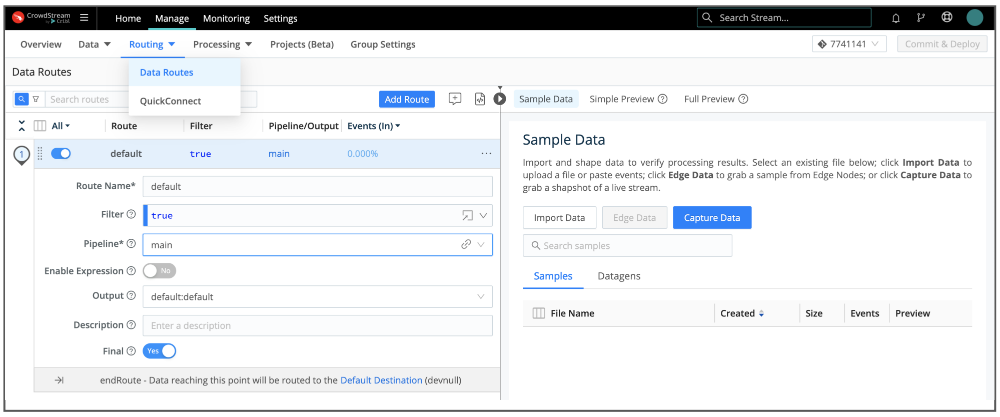
Checking Pipeline's Route connection

Configure a Route

In the Data Routes UI, a Route is where you complete a data-flow connection. Each Route connects exactly one Pipeline with exactly one Destination.

This combination can consume data from one, multiple, or all the Sources you've configured on the Data Routes side. (QuickConnect Sources are separate, but you can open those Sources' configs to move them to the Data Routes UI.) This is governed by the filtering we'll see below.

To open the Routing table, from a Group's **Manage** tab, select **Routing** > **Data Routes**.



Route	Filter	Pipeline/Output	Events (In)
default	true	main	0.000%

Route Name*	default
Filter	true
Pipeline*	main
Enable Expression	No
Output	default:default
Description	Enter a description
Final	Yes

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

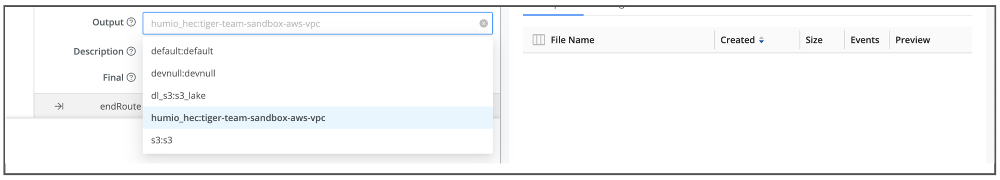
Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

Accessing the Routing table

As you can see, CrowdStream ships with one starter Route named **default**. It's configured as follows:

- The **Filter** expression is set to **true**. This simple, default expression makes data from **all** active Sources available to this Route. (See finer filtering options below.)
- The **Pipeline** drop-down is where you set the 1:1 Pipeline:Route connection. This starter Route's Pipeline defaults to **main**, but you can select any one other configured Pipeline.
- The **Output** drop-down defines the single Destination for this Pipeline:Route connection. If this isn't already set to your LogScale Destination, you'll most likely want to select that here. (Note that each option displayed on this list corresponds to the unique, arbitrary **Output ID** set on one **instance** of a Destination type. So some names might initially be unfamiliar.)



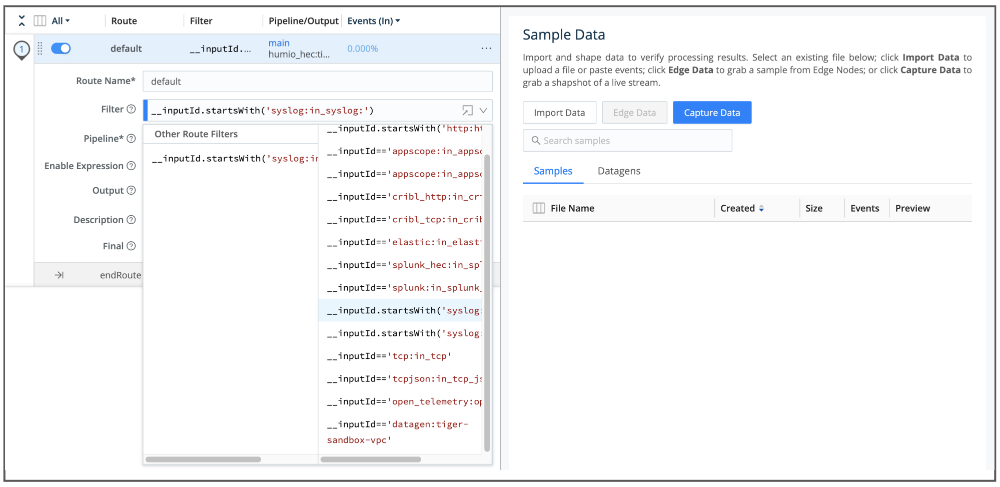
Changing a Route's Destination (Output)

To get data flowing, jump ahead to commit and deploy your new configuration.

Filter a Route's Input Data

When you want a Route to ingest data from only one Source, or a subset of Sources, you expand its **Filter** expression beyond the default **true** catch-all.

Filter is itself a drop-down. Open it to select among typeahead suggestions for several common inbound data types.



Route Name*: default

Filter:

```
__inputId.startsWith('syslog:in_syslog:')
```

```
__inputId.startsWith('syslog:in
```

```
__inputId.startsWith('http:ht
__inputId=='appscope:in_appsc
__inputId=='appscope:in_appsc
__inputId=='cribl_http:in_cr
__inputId=='cribl_tcp:in_crib
__inputId=='elastic:in_elast
__inputId=='splunk_hec:in_sp
__inputId=='splunk:in_splunk_
__inputId.startsWith('syslog
__inputId.startsWith('syslog
__inputId=='tcp:in_tcp'
__inputId=='tcpjson:in_tcp_js
__inputId=='open_telemetry:op
__inputId=='datagen:tiger-
sandbox-vpc'
```

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a shapshot of a live stream.

Tuning a Route's Filter expression

You can select one of these expressions and then modify it, using the same `__inputId` syntax. In the example expression below, note that the identifier after the `:` is – as with the **Output** options we saw above – the arbitrary name of one Syslog Source instance:

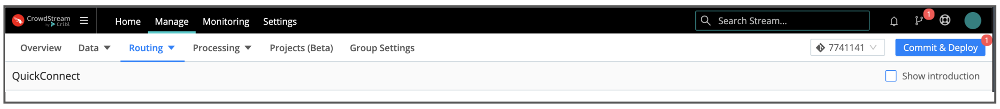
```
__inputId.startsWith('syslog:in_syslog:')
```

For More Information

- Cribl Expression Syntax > Filters

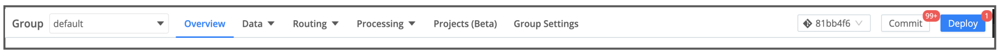
Commit/Deploy Config Changes

To get data flowing as you expect, you'll always need to use CrowdStream's built-in Git version-control integration to commit your configuration changes and deploy them to your event-processing Worker instances. After any significant configuration change, you'll be prompted with Group-level **Commit/Deploy** buttons like this at the upper right:



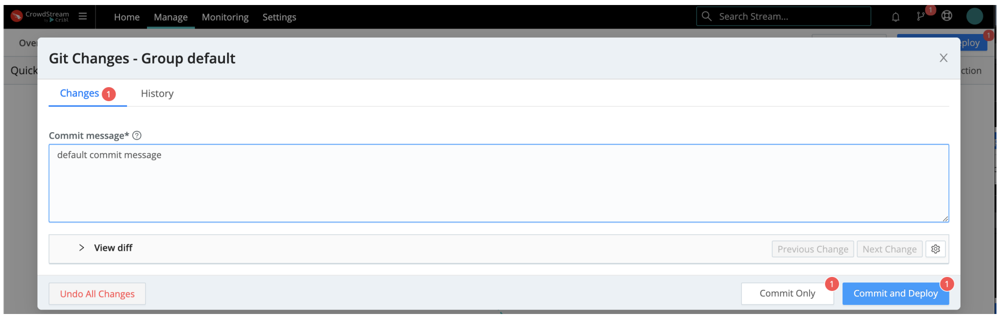
Commit & Deploy prompt

Depending on your version-control settings, **Commit** and **Deploy** might appear as separate buttons:



Commit and Deploy as separate buttons

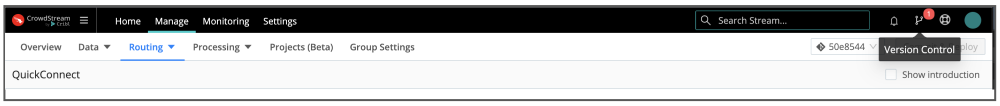
Clicking either flavor of the **Commit** button will display a confirmation modal like this. You can either accept the default commit message, or enter a specific message. Then be sure to select the appropriate buttons to both commit **and** deploy your changes.



Confirming a commit

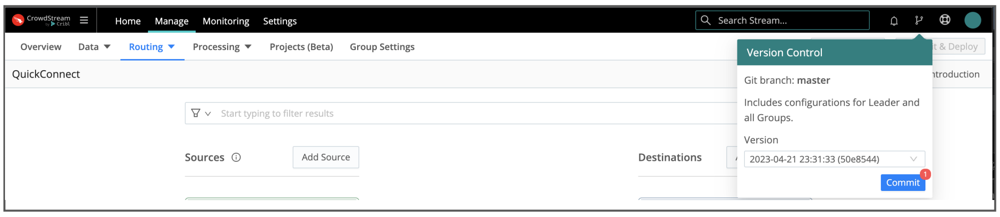
Data should now start flowing through the Workers. (You can check this by selecting the top nav's **Monitoring** link. This takes you to read-only dashboards, so remember to click **Manage** when you want to return to configuration options.)

In return for clearing all the red-dot prompts, your reward will be one more red dot, on the top nav's CrowdStream-wide Version Control button. This is because CrowdStream sees the preceding deploy to Workers as a change to record (commit) on the **Leader's** configuration.



Leader commit prompt

This top-level commit isn't necessary to enable data flow, but it backs up and safeguards your overall configuration. Click the button to display one final confirmation prompt:



Version Control

Git branch: master

Includes configurations for Leader and all Groups.

Version

2023-04-21 23:31:33 (50e8544)

Commit

Confirming a commit on the Leader

Click **Commit** here, and your configuration is complete!

For More Information

- Version Control > Committing Changes
- Version Control > Collapse Actions
- Monitoring

Moving Ahead with CrowdStream

To quote a fundamental treatise on DevSecOps: you know all the rules by now, and the fire from the ice. To delve deeper into CrowdStream's data shaping and routing options, follow these links to conceptual overview and reference topics.

For More Information

- Stream Basic Concepts
- Event Model

- Data Preview
- Functions
- Knowledge Objects
- Distributed Quick Start – This tutorial’s middle sections walk you through configuring Pipelines, Functions, and Routes, and scaling Workers, via the Routing UI
- Techniques & Tips
- Setup Guides
- API, Expressions, etc. Reference

4. Deploying Cribl Stream Software

4.1. Planning And Sizing

4.1.1. Deployment Planning

There are at least **three** key factors that will determine the type of Cribl Stream deployment in your environment:

- Amount of Incoming Data: This is defined as the amount of data planned to be ingested per unit of time. E.g., how many MB/s or GB/day?
- Amount of Data 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.

These factors are covered in detail in Sizing and Scaling, and in our Architectural Considerations introduction to reference architectures.

Type of Deployment

- Use Cribl.Cloud to quickly launch a Cribl-hosted deployment of the combined Cribl applications suite (Stream, Edge, and Search). With this option, Cribl assumes responsibility for provisioning and managing all infrastructure, on your behalf.
- Use Single-Instance/Basic Deployment when incoming data volume is low, and/or amount of processing is light.
- Use Distributed Deployment to accommodate increased load. (See Sizing and Scaling for detailed guidance. See Bootstrap Workers from Leader to streamline Workers' deployment via scripting.)

OS and System Requirements

Leader and Worker 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.

In the table below, we assume that **1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs)**. This corresponds to Intel/Xeon or AMD processors. On Graviton2/ARM64 processors, where 1 core is equivalent to 1 vCPU – but with higher capacity – sizing can be slightly different. For details, see Sizing and Scaling and Requirements.

Although the table shows only tested distro's, Cribl Stream's general requirements are 64-bit Linux kernel >= 3.10 and glibc >= 2.17.

Requirement Type	Requirements Details
Minimum Leader and Worker Nodes.	**OS:** Linux: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2 **System:** +4 physical cores, +8GB RAM, 5GB free disk space (more if persistent queuing is enabled on Workers)
Recommended Leader Node	**OS:** Linux: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2 **System:** +4 physical cores, +8GB RAM, 5GB free disk space
Recommended Worker Nodes	**OS:** Linux: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2 **System:** +8 physical cores, +32GB RAM, 5GB free disk space.

Browser Requirements

Most modern browsers will work, but Cribl Stream formally supports the five most-recent versions of Chrome, Firefox, Safari, and Microsoft Edge.

Port Requirements

See Ports for detailed information of ports which need to be open for Cribl Stream and its intergrations to work.

FIPS Mode Requirements

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

See the FIPS Mode topic for system and password requirements, and instructions for running in FIPS mode.

> Once Cribl Stream has been started without FIPS mode enabled, you cannot put it into FIPS mode. You must enable FIPS mode as described here, after installing **but before starting** Cribl Stream.

Cluster Installation/Configuration Checklist

This section compiles basic checkpoints for successfully launching a distributed cluster.

1. Provision Hardware

- 1 Leader Node (see specs/requirements in OS and System Requirements above).
- 4 Worker Nodes (see specs/requirements in OS and System Requirements above).
- Acquire an evaluation (Sales Trial) License from the Cribl Sales Team.

2. Configure Leader Node

- Install `git` if not present (e.g., `yum install git`).
- Open the necessary ports.
- Download, Install, and Launch Cribl.
- Enable Start at Boot.
- Configure as a Leader.
- Confirm Worker Processes Settings at `-2` (via **Settings** > **Global Settings** > **System** > **Manage Processes**).
- Install License.

3. Configure Worker Nodes

- Enable GUI Access. Administrators will need to connect to the TCP:9000 port on each Node.
- Download, Install, and Launch Cribl.
- Enable Start at Boot.
- Configure as a Worker. (Give each Worker the (arbitrary) tag `POV`.)

- Confirm Worker Processes Settings at `-2` (via **Settings** > **Global Settings** > **System** > **Manage Processes**).
- Install License.

4. Map Workers to Groups

- On the Leader Node, create a Worker Group.
  - Name the Worker Group (arbitrarily) `POV`.
- On the Leader Node, confirm that workers are connecting.
  - From the Leader Node's top menu, select **Workers**.
- Map Workers to `dev` Worker Groups.
  - Use the Filter: `cribl.tags.includes('POV')`.

5. Other

If you will be using Cribl Stream's GeoIP enrichment feature, install the MaxMind database onto the Cribl Stream Leader and all Worker Nodes.

4.1.2. Sizing And Scaling

A Cribl Stream installation can be scaled **up** within a single instance and/or scaled **out** across multiple instances. Scaling allows for:

- Increased data volumes of any size.
- Increased processing complexity.
- Increased deployment availability.
- Increased number of destinations.

We offer specific examples below, and introduce reference architectures in Architectural Considerations.

Scale Up

A Cribl Stream installation can be configured to scale up and utilize as many resources on the host as required. In a single-instance deployment, you govern resource allocation through the **Settings** > **Global Settings** > **System** > **Service Processes** > **Processes** section.

In a distributed deployment, you allocate resources per Worker Group. Navigate to **Manage** > `group-name` > **Group Settings** > **Worker Processes**.

Either way, these controls are available:

- **Process count**: Indicates the number of Worker Processes to spawn. Positive numbers specify an absolute number of Workers. Negative numbers specify a number of Workers relative to the number of CPUs in the system. like this: `{<number of CPUs available>` minus `<this setting>}`. The default is `-2`.
  Cribl Stream will correct for an excessive + or - offset, or a `0` entry, or an empty field. Here, it will guarantee at least the **Minimum process count** set below, but no more than the host's number of CPUs available.
- **Minimum process count**: Indicates the minimum number of Worker Processes to spawn. Overrides the **Process count**'s lowest result. A `0` entry is interpreted as "default," which here yields `2` Processes.
- **Memory (MB)**: Amount of heap memory available to each Worker Process, in MB. Defaults to `2048`. (See Estimating Memory Requirements below.) Heap memory is allocated dynamically from the OS as it is requested by the Process up to the amount dictated by this setting. If a Process needs more than the amount of heap memory allocated, it will encounter an `Out of memory` error, which is logged to `$CRIBL_HOME/log/cribl_stderr.log`. This error indicates either a memory leak or that the memory requirements are too low for the workload being performed by the Process. Cribl support can help you troubleshoot this situation.

For changes in any of the above controls to take effect, you must click the **Manage Processes** page's **Save** button, and then restart the Cribl Stream server via **Settings** > **Global Settings** > **System** > **Controls** > **Restart**. In a distributed deployment, also deploy your changes to the Groups.

For example, assuming a Cribl Stream system running on Intel or AMD processors with 6 physical cores hyperthreaded (12 vCPUs):

- If **Process count** is set to `4`, then the system will spawn exactly 4 processes.
- If **Process count** is set to `-2`, then the system will spawn 10 processes (12-2).

Allocate a Maximum Number of Worker Processes

To provision a container with a set amount of Worker Processes, use the `CRIBL_MAX_WORKERS` environment variable. Setting a maximum amount of Worker Processes helps prevent overloading the node with more CPUs than it can handle.

A couple notes about the `CRIBL_MAX_WORKERS` environment variable:

- The value you designate represents the **maximum** number of Worker Processes for the container. If there are fewer actual CPUs than designated in the variable, the container will provision with the number of Worker Processes that match the actual number of CPUs.
- Several different factors determine the final number of Worker Processes. Cribl uses the lowest number of all of these:
    - The Cribl license Worker Process limit
    - The number of actual Worker Processes (with **Process count** factored in)
    - The Worker count configured in `cribl.yml`
    - `CRIBL_MAX_WORKERS` (if configured)

  For example:
    - License limit: 7
    - Number of actual Worker Processes (OS based): 4, calculated as number of actual Worker Processes (`6`) and **Process count** (`-2`)
    - Worker Processes in `cribl.yml`: 5
    - `CRIBL_MAX_WORKERS`: 3

  There are three final assigned Worker Processes, because 3 is the lowest number of all the values above.

Example configurations

Docker

```
docker run -e CRIBL_MAX_WORKERS=2
```

Helm

```
helm install --set env.CRIBL_MAX_WORKERS=2
```

The `values.yml` file

```
  env:
    CRIBL_MAX_WORKERS: 2
```

Scale Out

When data volume, processing needs, or other requirements exceed what a single instance can sustain, a Cribl Stream deployment can span multiple Nodes. This is known as a distributed deployment. Here, you can centrally configure and manage Nodes' operation using one administrator instance, called the Leader.

It's important to understand that Worker Processes operate in parallel, i.e., independently of each other. This means that:

1. Data coming in on a single connection will be handled by a single Worker Process. **To get the full benefits of multiple Worker Processes, data should come in over multiple connections.**
   For example, it's better to have five connections to TCP 514, each bringing in 200GB/day, than one at 1 TB/day.
2. Each Worker Process will maintain and manage its own outputs. For example, if an instance with two Worker Processes is configured with a Splunk output, then the Splunk destination will see two inbound connections.
   For further details, see Shared-Nothing Architecture.

Capacity and Performance Considerations

As with most data processing applications, Cribl Stream's expected resource utilization will be proportional to the type of processing that is occurring. For instance, a Function that adds a static field on an event will likely perform faster than one that applies a regex to find and replace a string. Currently:

- A Worker Process will utilize up to 1 physical core (encompassing either 1 or 2 vCPUs, depending on the processor type.
- Processing performance is proportional to CPU clock speed.
- All processing happens in-memory.

- Processing does not require significant disk allocation.

Throughout these guidelines, we assume that **1 physical core** is equivalent to:

- 2 virtual/hyperthreaded CPUs (vCPUs) on Intel/Xeon or AMD processors.
- 1 vCPU on Graviton2/ARM64 processors.

Overall guideline: Allocate 1 physical core for each 400GB/day of IN+OUT throughput.

Examples

Estimating Number of Cores

In estimating core requirements – per processor type, it’s important to distinguish among vCPUs versus physical cores.

Intel/Xeon/AMD Processors with Hyperthreading Enabled

To estimate the number of cores needed: Sum your expected input and output volume, then divide by 400GB per day per physical core.

- Example 1: 100GB IN -> 100GB out to each of 3 destinations = 400GB total = 1 physical core (or 2 vCPUs).
- Example 2: 3TB IN -> 1TB out = 4TB total = 10 physical cores (or 20 VCPUs).
- Example 3: 4 TB IN -> full 4TB to Destination A, plus 2 TB to Destination B = 10TB total = 25 physical cores (or 50 vCPUs).

Graviton2/ARM64 Processors

Here, 1 physical core = 1 vCPU, but overall throughput is ~20% higher than a corresponding Intel or AMD vCPU. So, to estimate the number of cores needed: Sum your expected input and output volume, then divide by 480GB per day per vCPU.

- Example 1: 100GB IN -> 100GB OUT to each of 3 destinations = 400GB total = 1 physical core.
- Example 2: 3TB IN -> 1TB OUT = 4TB total = 8 physical cores.
- Example 3: 4 TB IN -> full 4TB OUT to Destination A, plus 2 TB OUT to Destination B = 10TB total = 21 physical cores.

Remember to also account for an additional core per Worker Node for OS/system overhead.

Estimating Number of Nodes

When sizing the number of Stream Worker Nodes, there are two other factors to consider:

- Number of Nodes sized for peak workloads;
- Number of Nodes offline.

This will allow you to create a robust Cribl Stream environment that can handle peak workloads even during rolling restarts and maintenance downtime.

General guidance includes:

- On each Node, Cribl recommends at least 4 ARM vCPUs, or at least 8 x86 vCPUs (i.e., 4 cores with hyperthreading). Below this theshold, the OS overhead from reserved threads claims an excessive percentage of your capacity.
- On each Node, Cribl recommends no more than 48 ARM vCPUs, or 48 x86 vCPUs (i.e., 24 cores with hyperthreading). This allocation handles disk I/O requirements when persistent queueing engages.
- Plan to successfully handle peak workloads even when 20% of your Worker Nodes are down. This allows for OS patching, and for conducting rolling Stream upgrades and restarts.
- For customers in the 5–20 TB range, size for 4-8 Worker Nodes per Worker Group.

Sizing Example for a Deployment

Step 1: Calculate the total inbound + outbound volume for a given deployment.

In this example, your deployment has 6TB/day streaming into Worker Nodes for a given site, and this data is being routed to both cheap mass storage (S3) and to your system of analysis – 10 TB/day going out.

Stream Workers must have enough CPUs to handle a total of 16 TB per day IN+OUT.

Step 2: From the available server configurations, size the correct number of Nodes for your environment.

You are considering an AWS Compute-optimized Graviton EC2 instance, with either 16-vCPU or 8-vCPU options. We’ll walk you through how to calculate based on each option.

Using Graviton 16-vCPU instances

- Each Node would have 15 vCPUs available to Cribl Stream, each at 480 GB per day per vCPU.
- 15 vCPUs per Node at 480 GB per day per CPU = 7200 GB/day per Node.
- 7200 GB/day divided by 1024 GB/TB = 7 TB/day.
- To handle the workload of 16 TB/day, you need 3 Nodes. Make sure you always round up.

- 20% of 3 Nodes ≈ 1 Node. Make sure you always round up. To provide for redundancy during patching, maintenance, or restarts, you need one additional server.

With 4x 16-vCPU Nodes, you have 28 TB/day capacity, and can support 21 TB/day with one Node down.

Using Graviton 8-vCPU instances

- Each Node would have 7 vCPUs available to Stream, each at 480 GB per day per vCPU.
- 7 vCPUs per Node at 480 GB per day per CPU = 3360 GB/day per Node.
- 3360 GB/day divided by 1024 GB/TB = 3.3 TB/day.
- To handle the workload of 16 TB/day, you need 5 Nodes. Make sure you always round up.
- 20% of 5 Nodes = 1 Node. To provide for redundancy during patching, maintenance, or restarts, you need one additional server.

With 6x8-vCPU Nodes, you have 19.8 TB/day capacity, and can support 16.5 TB/day with one Node down.

If you are optimizing for price, AWS pricing is linear based on the total number of cores. So six 8-vCPU systems are roughly 75% the cost of four 16-vCPU systems.

If you are optimizing for potential future expansion, the 4x16-vCPU systems would offer ideal additional capacity.

Estimating Memory Requirements

The general guideline for memory allocation is to start with the default 2048 MB (2 GB) per Worker Process, and then add more memory as you find that you're hitting limits.

Memory use is consumed per component, per Worker Process, as follows:

1. Lookups are loaded into memory.
   (Large lookups require extra memory allocation – see Memory Sizing for Large Lookups.)
2. Memory is allocated to in-memory buffers to hold data to be delivered to downstream services.
3. Stateful Functions (Aggregations and Suppress) consume memory proportional to the rate of data throughput.
4. The Aggregations Function's memory consumption further increases with the number of **Group by**'s.
5. The Suppress Function's memory use further increases with the cardinality of events matching the **Key expression**. A higher rate of distinct event values will consume more memory.

Recommended AWS, Azure, and GCP Instance Types

You could meet the requirement above with multiples of the following instances:

AWS – Intel processors, Compute Optimized Instances. For other options, see here.

Minimum	Recommended
c5d.2xlarge (4 physical cores, 8vCPUs) c5.2xlarge (4 physical cores, 8vCPUs)	c5d.4xlarge or higher (8 physical cores, 16vCPUs) c5.4xlarge or higher (8 physical cores, 16vCPUs)

AWS – Graviton2/ARM64 processors, Compute Optimized Instances. For other options, see here.

Minimum	Recommended
c6g.2xlarge (8 physical cores, 8vCPUs) c6gd.2xlarge (8 physical cores, 8vCPUs)	c6g.4xlarge or higher (16 physical cores, 16vCPUs) c6gd.4xlarge or higher (16 physical cores, 16vCPUs)

Azure – Compute Optimized Instances

Minimum	Recommended
Standard_F8s_v2 (4 physical cores, 8vCPUs)	Standard_F16s_v2 or higher (8 physical cores, 16vCPUs)

GCP – Compute Optimized Instances

Minimum	Recommended
c2-standard-8 (4 physical cores, 8vCPUs) n2-standard-8 (4 physical cores, 8vCPUs)	c2-standard-16 or higher (8 physical cores, 16vCPUs) n2-standard-16 or higher (8 physical cores, 16vCPUs)

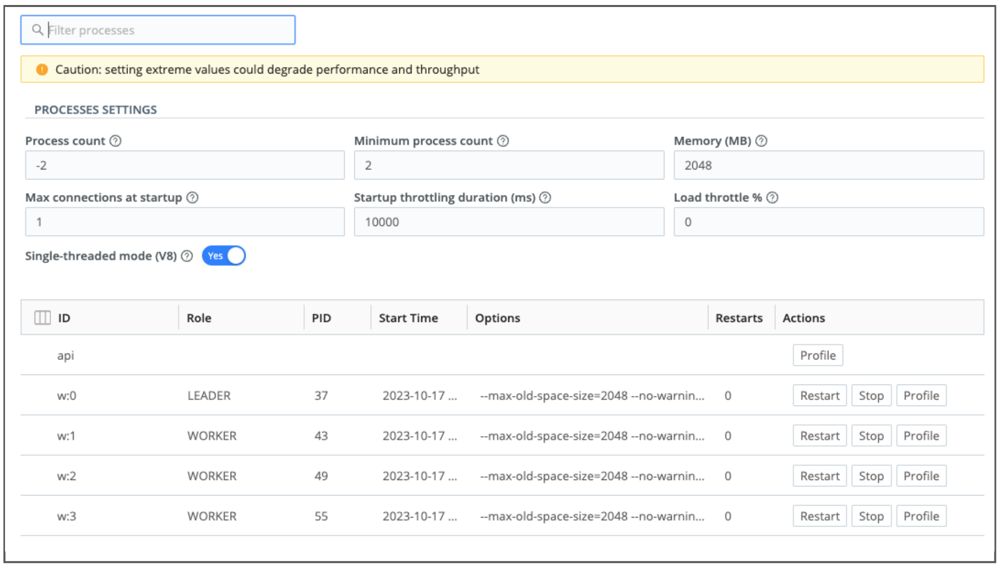
In all cases, reserve at least 5GB disk storage per instance, and more if persistent queuing is enabled.

Measuring CPU or API Load

You can profile CPU and API usage on individual Worker Processes.

Single-Instance Deployment

Go to **Settings** > **Global Settings** > **System** > **Service Processes** > **Processes**, and click **Profile** on the desired row.


Caution: setting extreme values could degrade performance and throughput

PROCESSES SETTINGS

Process count: -2	Minimum process count: 2	Memory (MB): 2048
Max connections at startup: 1	Startup throttling duration (ms): 10000	Load throttle %: 0

Single-threaded mode (V8): Yes

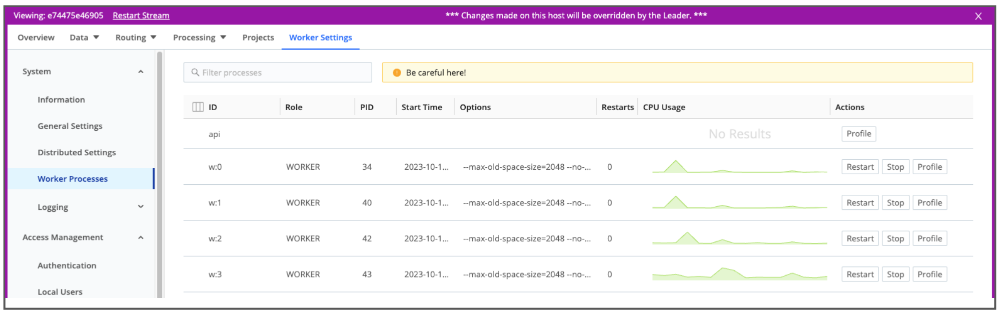
ID	Role	PID	Start Time	Options	Restarts	Actions
api						Profile
w:0	LEADER	37	2023-10-17 ...	--max-old-space-size=2048 --no-warnin...	0	Restart Stop Profile
w:1	WORKER	43	2023-10-17 ...	--max-old-space-size=2048 --no-warnin...	0	Restart Stop Profile
w:2	WORKER	49	2023-10-17 ...	--max-old-space-size=2048 --no-warnin...	0	Restart Stop Profile
w:3	WORKER	55	2023-10-17 ...	--max-old-space-size=2048 --no-warnin...	0	Restart Stop Profile

Worker CPU profiling (single-instance)

Distributed Deployment

This requires a few more steps:

1. Enable Worker UI Access if you haven't already.
2. From the top nav, select **Manage** > **Workers**.
3. Click on the **GUID** link of the Worker Node you want to profile. (You will now see that GUID in a **Worker** drop-down at the top left, above a purple header that confirms that you've tunneled through to the Worker Node's UI.)
4. Select **Worker Settings** from that Worker Node's top nav.
5. Select **System** > **Worker Processes** from the resulting side nav.
6. Click **Profile** on the desired Worker Process. To profile the API usage, click **Profile** on the **api** row.


Viewing: e74475e46905 Restart Stream
*** Changes made on this host will be overridden by the Leader. ***
Overview Data Routing Processing Projects Worker Settings
System
Information
General Settings
Distributed Settings
Worker Processes
Logging
Access Management
Authentication
Local Users
Filter processes
Be careful here!

ID	Role	PID	Start Time	Options	Restarts	CPU Usage	Actions
api						No Results	Profile
w:0	WORKER	34	2023-10-1...	--max-old-space-size=2048 --no-...	0		Restart Stop Profile
w:1	WORKER	40	2023-10-1...	--max-old-space-size=2048 --no-...	0		Restart Stop Profile
w:2	WORKER	42	2023-10-1...	--max-old-space-size=2048 --no-...	0		Restart Stop Profile
w:3	WORKER	43	2023-10-1...	--max-old-space-size=2048 --no-...	0		Restart Stop Profile

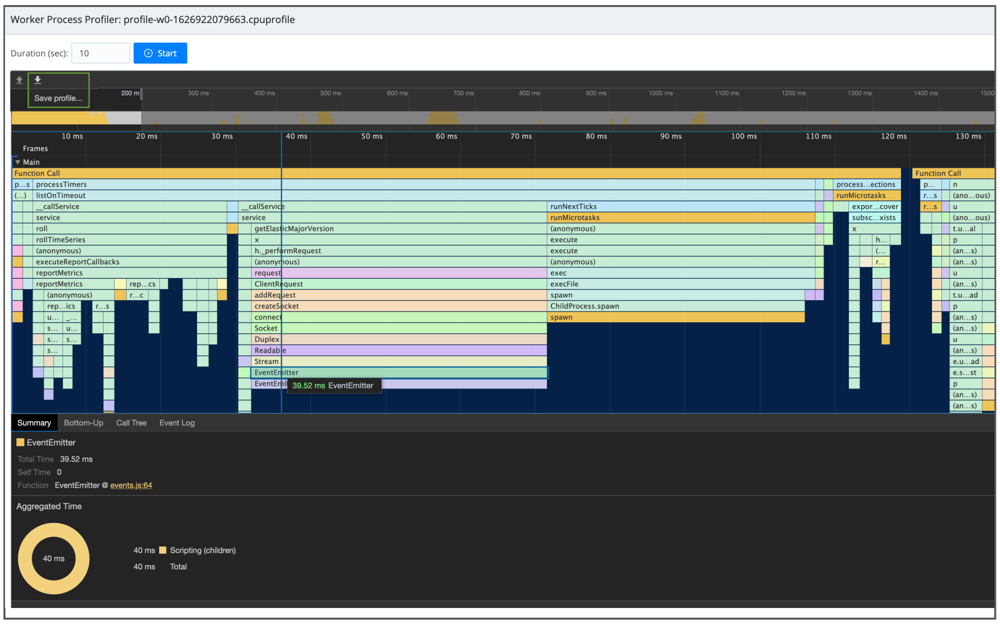
Worker CPU profiling (distributed)

Generating a CPU or API Profile

In either a single-instance or distributed deployment, you will now see a **Worker Process Profiler** modal.

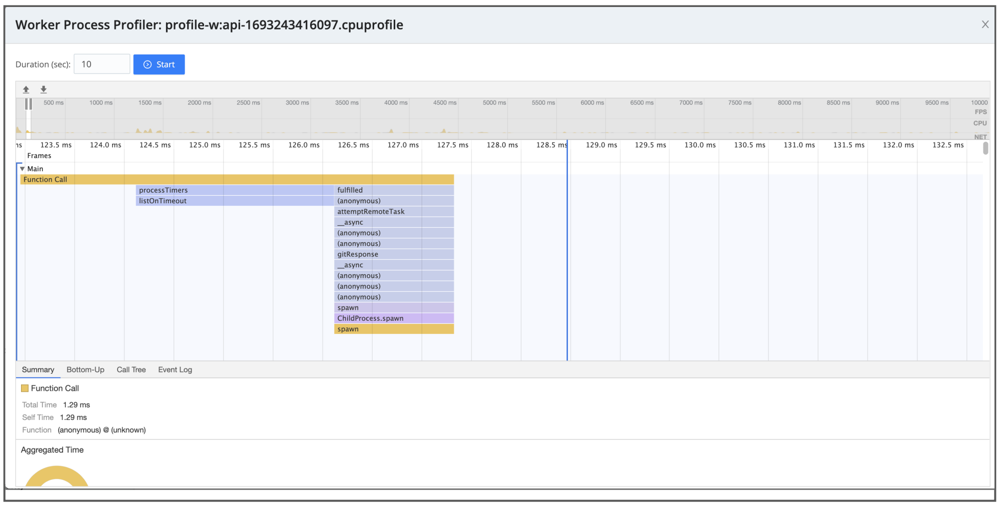
The default **Duration (sec)** of `10` seconds is typically enough to profile continuous issues, but you might need to adjust this – up to several minutes – to profile intermittent issues. (Longer durations can dramatically increase the lag before Cribl Stream formats and displays the profile data.)

Click **Start** to begin profiling. After the duration you've chosen (watch the progress bar), plus a lag to generate the display, you'll see a profile something like this:



Worker CPU profile

Profiling the API process can be a helpful troubleshooting tool if the Leader is slow.



API profile

Below the graph, tabs enable you to select among **Summary**, **Bottom-Up**, **Call Tree**, and **Event Log** table views.

To save the profile to a JSON file, click the **Save profile** ( ) button we've highlighted at the modal's upper left.

Whether you've saved or not, when you close the modal, you'll be prompted to confirm discarding the in-memory profile data.

See also: Diagnosing Issues > Including CPU Profiles.

4.1.3. Architectural Considerations

This guide introduces Cribl Stream reference architectures by outlining some considerations to keep in mind when architecting your deployment. It covers different potential architectures, with each approach's advantages, limitations, and alternatives.

Architecting a deployment is always about finding the balance between cost versus risk. Once you're familiar with the contents of this guide, you'll be able to consciously make decisions about where your deployment should sit on that spectrum.

See our reference architectures for different use cases. You can replay several Webinar presentations of this material on Cribl's YouTube channel.

"It Depends"

This guide is not intended to replace Sizing and Scaling, but rather to complement it by outlining the consequences of various architectural decisions. (As needed, please either read Sizing and Scaling first, or refer to it to look up concepts.)

Another great resource is Cribl's Sizing Calculator, where you can play with the numbers as you're considering different deployment options.

General Sizing Considerations

Some basic sizing variables are your processors, expected data throughput, and requirements for memory, disk space, connections volume, and the Stream Leader's own needs.

Processors

Your primary consideration in sizing processors is your data throughput capacity, which depends on the number of Worker Processes.

Overall guideline: Allocate 1 physical core for each 400GB/day of IN+OUT throughput.

This can vary based on processor speeds or types. (For example, Graviton ARM processors' throughput is 20% higher than x86-64 processors' reference 400GB/day.) But in this guide's calculations, we'll assume 400GB/day IN+OUT per processor, and dig into what that really means. Please adjust your own calculations as needed.

Throughput

Where we reference total throughput capacity below: Because Cribl Stream can send events to multiple Destinations and filter out unneeded events, IN won't always be the same as OUT. For example, 400GB IN+OUT could mean 200GB IN + 200GB OUT, but it could also mean 300GB IN + 100GB OUT, or any combination of IN+OUT that adds up to 400GB/day.

Memory

Memory requirements also play into architecture considerations. Depending on what type of processing you'll be doing, those requirements might vary. You might know your memory requirements in advance. Or you might monitor memory usage and increase it as your processing complexity increases. The default of 2GB per Worker Process is a good starting point.

Disk Space

The base recommendation, 5GB of free disk space, applies to the volume where Cribl Stream is installed. Beyond that, if you choose to use Persistent Queues, you will need to calculate additional disk space – based on the velocity of your data, and your desired duration before reaching a full queue. Normally, you must define the maximum disk space a queue can use, per Worker Process, for each Source or Destination. However, if you use a separate volume for Persistent Queues, you can remove those limits, and allow the queues to take up the whole volume if needed.

Object-store Destinations use a staging directory to write out files before sending them out. If you're using one of these Destinations, please review our Amazon S3 Better Practices for sizing and architecture considerations.

In both cases, using fast disks is important, to prevent the disks from becoming bottlenecks.

Number of Connections

Beyond throughput and pipeline complexity, some resources get used for setting up and breaking down TCP connections. In the field, Cribl often sees one CPU core support 400 connections (or 200 for vCPUs), but that depends on EPS (events per second) and other demands on the system resources.

If your calculations show more than 250 connections per CPU core, you should look closer at EPS and processing complexity. In our testing, once you hit the higher end of the EPS spectrum, 150 connections per core at 100 EPS still leave almost 25% capacity in reserve for spikes.

When it comes to outgoing connections, you should also consider memory utilization if you have a lot of TCP receivers. This is because each Worker Process makes a separate connection and consumes 2-4MB of RAM. Some Destinations let you adjust the maximum number of connections.

Leader Node

All the above discussion concerns Worker Nodes, because they are the ones responsible for processing the data. The Leader Node's requirements should not need to expand beyond these initial recommendations, unless the Leader needs to manage Edge Fleets, in which case reference Cribl's Edge docs.

Finding the Balance Between Risk and Cost

Beyond your basic requirements, you should of course plan contingency capacity to support resource demands that exceed ideal circumstances.

Plan for Failure

Given the wrong circumstances, failure will happen. Your job, when architecting a deployment, is to examine the various failure scenarios and decide what level of resources you are willing to allocate to prevent failure. This is analogous to deciding how many nines of reliability you're willing to pay for.

Throughput Variability

The 400GB/day IN+OUT guidance mentioned above assumes that your data flow is relatively flat. This means you're expecting that every second, each Worker Process is handling about ~4.74MB of data IN+OUT. However, in reality, the load varies based on time of day or other system activities.

For example, you might experience higher load during business hours. Or you might have an hourly upstream process that causes spikes of activity. Also, depending on the nature of your organization, you might have certain days, or planned events, when you expect users' increased load on your systems. Then there are unplanned events, such as outages or DDoS attacks, that might lead to decreased capacity or increased demands.

Another factor to consider, if you're using Persistent Queues, is that flushing those queues will increase the total throughput demand – including on the downstream systems that need to be able to handle the burst. Unfortunately, there are too many variables to have a generic recommendation here, but a starting point of ~10% headroom is a good idea. As we discuss in the conclusion, you'll need to test these assumptions.

Now that we're digging deeper into throughput, you should start thinking about where to set a threshold, beyond which you know your system is likely to fail. If you have data flowing into a system of analysis that can let you look at your current ingestion rate, this can be a good start.

You want to look at the busiest times, and drill down far enough to see what your historical maximum per-second throughput has been. Past performance is not guarantee of future results, but it will at least give you an idea of what to expect.

If you have new Sources, or if you don't have a good way of estimating current throughput, you'll have to make some assumptions about future behavior. In the end, take whatever you come up with for expected MB/sec throughput, add whatever safety margin you feel is appropriate, and multiply back out by 84 (24*60*60/1024) to get GB/day throughput.

Exceeding Capacity

As you are deciding how much safety margin to plan into your environment, it also helps to think through what happens when you exceed capacity. Depending on the types of Sources you have, an overloaded deployment might have different failure modes. This ranges from the possibility of immediate data loss in the case of UDP, to simple delays in the case of Pull Sources. Some Push Sources, such as Cribl Edge or Splunk Universal Forwarders, have the capacity to pause or buffer the data flow for some duration.

Single-Instance Deployment

The simplest type of deployment is single-instance. Let's look at the case of a 4 core (8 vCPUs), 16GB RAM, single-instance deployment:

Resource	Total	Used by WPs	Used by system/API
Cores (vCPUs)	4 (8)	3 (6)	1 (2)
RAM	16GB	12GB	4GB

With the default assumption of 400GB/day IN+OUT, and sending to a single Destination, the above configuration can process an average of 1200GB/day.

Let's make a few more assumptions:

- Peak usage during the day is 2x average.
- Persistent Queues for 4 hours of data.
- We want to withstand a 2x burst over typical usage from the Source.
- Our Source is syslog UDP*, so going over capacity means lost data.

(* We're assuming that the load is distributed evenly among all Worker Processes. Further down, we'll talk about why we're making that assumption, and the dangers of "pinning" Worker Processes.)

We're working backwards for this one, but it's a good exercise. Taking the 1200GB/day capacity, dividing it by 2 for peak load, and then dividing by 2 again for burst tolerance, we come up with 300GB/day adjusted capacity. So with this deployment, we can safely say we can handle a 300GB/day stream of syslog UDP data.

Because we want the ability to keep 4 hours of data in a Persistent Queue, we want to use the peak-load number for that. Our upfront assumption is that peak load is 2x normal load, which is 600GB/day, or 25GB/hour. 100GB of disk space should be sufficient, unless we expect the 2x bursts to last a significant duration. If we want to withstand a 1-hour burst during peak times, 125GB of disk space would be needed.

Where Single-Instance Fails

The obvious practical limitation here is that all data is flowing through a single system. This means that if you bring your single-instance deployment down for patching, or experience an unexpected outage, you'll be losing data. This might be acceptable for test purposes, but in production, there are very few scenarios in which you **won't** lose data with a single-instance deployment:

- All your Sources are Pull Sources.
- Your Push Sources can pause the data flow when Stream Workers are not available.

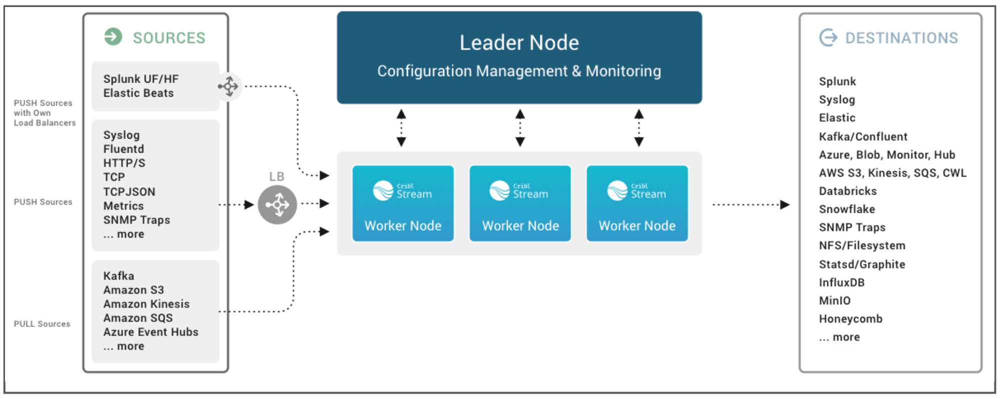
A distributed deployment, with multiple Worker Nodes, is a more robust solution that doesn't have these limitations. So next, let's look at a simple distributed example.

Distributed Deployment

The simplest distributed deployment involves a Leader Node and one or more Worker Nodes. A single Worker Node suffers from the same limitations as the single-instance deployment, so in this section we'll look at scenarios with multiple Worker Nodes.

You might be sizing a Worker Group when working with either a Cribl.Cloud Leader or an on-prem Leader. For an on-prem Leader, use our recommended system requirements.

The main thing to consider with multiple Workers is that you'll need a load balancer for all Push Sources that don't natively support balancing across multiple Worker Nodes. You can see this added load balancer called out (as **LB**) in our Distributed Deployment Architecture diagram:



Distributed deployment architecture

How Many Worker Nodes?

The next consideration is, how many Worker Nodes do you need? Having multiple Worker Nodes allows you to continue processing data while one Node is down for maintenance – so your sizing is determined by the remaining Worker Nodes. Here, you have a matrix of adjusting scaling vertically vs. horizontally, so let's look at the math:

Effectively, 2 Worker Nodes means you lose 50% capacity when one is down. With 3 Nodes, you lose 33%. With 6 Nodes, you lose 17%. You will lose 2 vCPUs in overhead on each Node, but at larger scales, the benefits outweigh the costs.

Let's look at an example using the naive 400GB/day per core capacity (i.e., 200GB/day per vCPU or Worker Process). We'll pick approximately comparable desired throughput of 5.6-6TB/day, and see how the math works out.

2 Worker Nodes

32 vCPUs x2 Nodes = 64 vCPUs (- 2 vCPUs/Node for overhead) = 60 WPs

1 Node downs during patching: 30 WPs*0.2=6 TB capacity

6 TB/64 vCPUs = 0.09 TB/vCPU

3 Worker Nodes

16 vCPUs x3 Nodes = 48 vCPUs (-2 vCPUs/Node for overhead) = 42 WPs

1 Node downs during patching: 28 WPs*0.2=5.6 TB capacity

5.6 TB/48 vCPUs = 0.12 TB/vCPU

6 Worker Nodes

8 vCPUs x6 = 48 vCPUs (-2 vCPUs/Node for overhead) = 36 WPs

1 Node downs during patching: 30 WPs*0.2 TB/WP = 6 TB capacity

6TB/40 vCPUs = 0.15 TB/vCPU

As we can see, once your required throughput is high enough to make the 2 vCPU/Node tax negligible, a larger number of Worker Nodes is advantageous. You can do the math for your specific use case – remembering to adjust for expected peak usage and desired spike tolerance.

That said, having **3 Worker Nodes is usually the recommended minimum**, since this provides resilience against unplanned outages even when one Worker is down for planned maintenance.

Where a Single Worker Group Can Fail

Even provisioning an “adequate” number of Worker Nodes, in a single Worker Group, might not be fully resilient if you encounter connection pinning, data bursts, or other unexpected resource demands.

Pinning

In Single-Instance Deployment above, we included the caveat of “assuming that the load is distributed evenly among all Worker Processes.” In reality, you need to validate this assumption. In some situations, you’ll need to take additional measures to assure that it’s true, or work around cases when it is not.

Pinning is a situation where a single TCP or UDP Source is sending a large amount of data over a single connection, causing that connection to be “pinned” to a single Worker Process. If the throughput is high enough, and the load balancer is unable to break up that connection, the affected Worker Processes might be overloaded and unable to keep up.

This situation is best resolved by having the sender establish separate connections. If that’s not possible, try isolating a Worker Group that receives this input from all other processing responsibilities – this will maximize the throughput that this Group can handle.

Burst Loads Affecting Push Sources

Most of the time, your Push Sources will generate a reasonably predictable, consistent load. Using the sizing considerations outlined above, you should be able to come up with a Worker Group sized appropriately for your desired risk-versus-cost balance.

However, this balance can be upset when you have a surge of data that makes your system temporarily busier than you accounted for. The greatest impact would be felt by Push Sources that have no way to handle backpressure, such as UDP Sources.

A prime example of data burst is an ad-hoc run of a large Collection job. With multiple Worker Groups, you can separate Push versus Pull Sources, and can even use an Auto Scaling group group for large Collection jobs if desired.

Unexpected Resource Hogs

Our throughput estimates are based on certain assumptions about the average level of complexity of operations performed on the data. As you’re creating your Pipelines, at some point you might decide that you require additional resources to meet your data-processing needs. Certain combinations of Function, logic, and throughput will consume extra resources.

When expanded resource demands happen in a predictable, controlled manner, all is well. However, we've seen cases where an inefficient regular expression, or a memory-hungry custom Collection script, will cause a whole Worker Group to get overloaded. This leads to crashes, or to performance poor enough to cause data loss. Pipeline profiling helps with some of this. But using separate Worker Groups is an option to provide additional isolation.

Multiple Worker Groups

At the beginning of Distributed Deployment, we considered the limitations of a single Worker Group, and examined some strictly sizing-related reasons for multiple Worker Groups. Here, let's consider some other reasons you might decide to take advantage of multiple Worker Groups.

In some cases, it makes sense to create separate Worker Groups, each close to their data. In other cases, multiple Groups might be necessary for your architecture based on other needs. Here are a few examples, which might not be directly applicable to your scenario, but which should help you realize when multiple Groups might benefit your deployment:

- Keeping Workers close to their data to optimize syslog over UDP.
- Controlling egress costs, by reducing and compressing data before it leaves your cloud provider's region.
- Separating Push Sources versus Pull/Collector Sources.
- Geographical restrictions on data location, such as GDPR.
- Reducing or increasing the number of Workers connecting with a Source or a Destination.
- Centralize certain processing that is dependent on external systems, such as Redis lookups.
- Reducing the frequency of Stream config deployments, in cases where a deployment might disrupt data flow – such as triggering rebalancing among Kafka-based Sources.

Conclusion

As you can see by now, our “it depends” subtitle applies. But we hope that reading this guide has familiarized you with key aspects of what “it” depends on. There is no single “best practice” for sizing, because your own best architecture will depend on considering all options, including your risk-versus-cost balance, to define your own solution.

Equipped with the contents of this guide, you should be able to make better decisions about the potential risks and mitigations appropriate for your environment. After initial deployment, you can monitor utilization trends in the different resources we covered. As your needs evolve, so can your architecture. You'll make assumptions, and when the potential for failure is beyond your tolerance, you can adjust accordingly.

Some final considerations: Is it worth stress-testing your environment, or a downscaled version of it? What’s the likelihood of everything going wrong at the same time – your throughput volume spiking to the highest acceptable level, while your Destinations are unavailable and your Persistent Queues are filling up? It depends.

To firm up your own architecture, consult Cribl support, Community Slack, or the Cribl Solutions Engineering or Professional Services experts dedicated to your deployment.

4.2. Manual Deployment

4.2.1. Deployment Types

Deployment guide to get you started with self-hosted Cribl Stream

---

There are at least **two** key factors that will determine the type of Cribl Stream deployment in your environment:

- Amount of Incoming Data: This is defined as the amount of data planned to be ingested per unit of time. E.g. How many MB/s or GB/day?
- Amount of Data Processing: This is defined as the amount of processing that will happen on incoming data. E.g., is most data passing through and just being routed? Or are there a lot of transformations, regex extractions, field encryptions? Is there a need for heavy re-serialization?

Single-Instance Deployment

When volume is low and/or amount of processing is light, you can get started with a single instance deployment.

Distributed Deployment

To accommodate increased load, we recommend scaling up and perhaps out with multiple instances.

Splunk App Deployment

If you have an existing Splunk Heavy Forwarder infrastructure that you want to use, you can deploy Cribl App for Splunk. See the note below before you plan.

Cribl App for Splunk Deprecation Notice

Please see details here.

Kubernetes/Helm Deployment

You can deploy Cribl Stream Leader Nodes (or single instances) and Worker Nodes via Cribl's Helm charts.

Docker Deployment

You can deploy Cribl Stream instances using images from Cribl's public Docker Hub.

Shared-Nothing Architecture

All Cribl Stream deployments are based on a shared-nothing architecture pattern, where instances/Nodes and their **Worker Processes operate separately, serving all inputs, outputs, and event processing independently of each other**.

This allows the overall system to continue to operate even if individual Processes or Nodes fail. It also allows individual Nodes to upgrade without system-wide downtime.

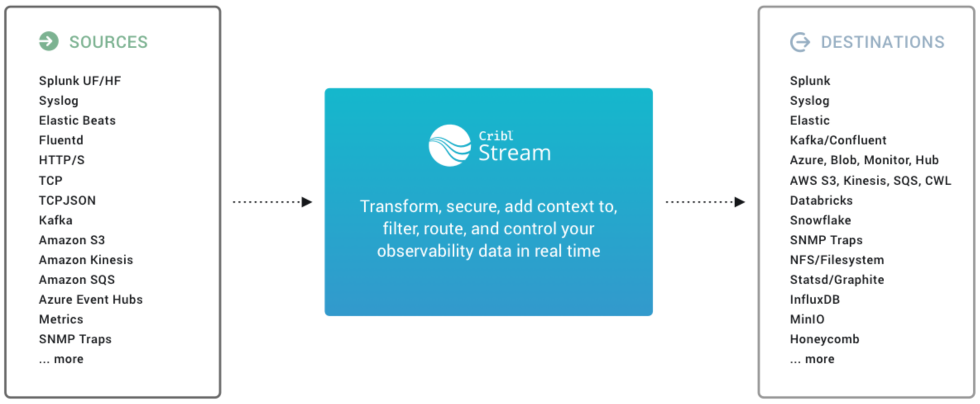
4.2.2. Single-Instance/Basic Deployment

Getting started with Cribl Stream on a single instance

For small-volume or light processing environments – or for test or evaluation use cases – a single instance of Cribl Stream might be sufficient to serve all inputs, event processing, and outputs. This page outlines how to implement a single-instance deployment.

This page also provides system requirements and startup procedures for a distributed deployment.

Architecture



Requirements

OS

- Linux kernel >= 3.10 and glibc >= 2.17.

Example distributions: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2.

Tested so far on Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2.

System

- 1 GHz (or faster), 64-bit CPU.
- 4+ physical cores, 8 GB+ RAM – all beyond your basic OS/VM requirements.

- 5 GB free disk space (more if persistent queuing is enabled).

Browser

- The five most-recent versions of Chrome, Firefox, Safari, and Microsoft Edge.

SELinux Support

Here, `enforcing` mode is supported, but not required.

All quantities listed above are minimum requirements. To fulfill these requirements using cloud-based virtual machines, see Recommended AWS, Azure, and GCP Instance Types.

System Requirements Details

We assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs) on Intel/Xeon or AMD processors; and to 1 (higher-throughput) vCPU on Graviton2/ARM64 processors. For additional details, see Estimating Number of Cores.

Your total memory allocation per host must accommodate all Worker Processes' memory usage, plus host OS requirements. Each Worker Process might use up to the maximum heap size, plus some Node.js overhead that isn't part of heap, plus a third memory draw that will scale upward with configuration details like your type and number of Destinations. Always monitor your Nodes' memory usage – and especially, check for new requirements after configuration changes like adding new Destinations. For additional details, see Estimating Memory Requirements.

Setting the `CRIBL_HOME` Environment Variable

The `CRIBL_HOME` env is available in the Cribl Stream application, but not on your terminal. If you want to use `$CRIBL_HOME`, you can:

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

FIPS Mode Requirements

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

See the FIPS Mode topic for system and password requirements, and instructions for running in FIPS mode.

Installing on Linux

- Install the package on your instance of choice. Download it here.
- Ensure that required ports are available (see Network Ports).
- Un-tar in a directory of choice, e.g., in the `/opt/` directory: `tar xvzf cribl-<version>-<build>-<arch>.tgz`

Installing Cribl Stream and Cribl Edge on the Same Host

You can run an Edge Node on a Cribl Stream Leader Node, or an Edge Node and a Worker Node on the same host. For details, see Installing Cribl Edge and Cribl Stream on the Same Host.

Running

To run Cribl Stream in FIPS mode, **do not** use the commands below right away; instead, first consult this topic.

Go to the `$CRIBL_HOME/bin` directory, where the package was extracted (e.g.: `/opt/cribl/bin`). Here, you can use `./cribl` to:

- **Start**: `./cribl start`
- **Stop**: `./cribl stop`
- **Reload**: `./cribl reload`
- **Restart**: `./cribl restart`
- **Get status**: `./cribl status`
- **Switch a distributed deployment to single-instance mode**: `./cribl mode-single` (uses the default address:port `0.0.0.0:9000`)

Executing the `restart` or `stop` command cancels any currently running collection jobs. For other available commands, see CLI Reference.

Next, go to `http://<hostname>:9000` and log in with default credentials (`admin:admin`). You can now start configuring Cribl Stream with Sources and Destinations, or start creating Routes and Pipelines.

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

Shutdown and Restart Sequence

When a Worker Process receives an explicit shutdown command, it follows this sequence:

1. Shuts down internal system communications: stops receiving any commands from the API Process or distributed Leader.
2. Shuts down the input Sources.
3. When the input stream ends, receives a signal event from Cribl Stream's event processor to flush out any stateful Pipeline Functions (such as Aggregations, Sampling, Dynamic Sampling, and Suppress).
4. Waits for 10 seconds, to allow data to finish flowing through the streams processing engine. This wait is designed to allow all Destinations to flush out remaining data. However, any data not flushed within this interval – e.g., because of an error on downstream receivers – will be lost.
5. Exits.

Shutdown/Restart with PQ

Enabling Persistent Queues, on Destinations that support it, generally helps ensure data delivery to your downstream systems. However, note that when a Worker Process restarts, there is a potential for duplicate events to be sent through such Destinations.

This is because PQ doesn't mark events as safe to discard until they've been handed them off to the host OS to send out. So if the Worker Process exits at the final step above **before** all events have flushed, the final handful of events will not have been marked as committed and re moved. Upon restart, Cribl Stream will still see them, and will resend them.

Enabling Start on Boot

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

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

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

Most Linux distributions now use `systemd` to start processes at boot, while older distributions might still use `initd`. If you are not sure which service should be configured at startup, check with your Linux administrator. Then follow the corresponding procedure below.

Using systemd

To **enable** Cribl Stream to start at boot time with **systemd**, you need to run the `boot-start` command. Make sure you first create any user you want to specify to run Cribl Stream. E.g., to run Cribl Stream 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.service`, and will start Cribl Stream 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 Stream 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
```

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
```

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

Installed systemd File

```
[Unit]
Description=Systemd service file for Cribl Stream.
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 on systemd

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

```
systemctl edit cribl
```

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

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

Do NOT Run Cribl Stream as Root!

If Cribl Stream needs to listen on low ports 1–1024, it will need 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
```

Using initd

To **enable** Cribl Stream to start at boot time with **initd**, you need to run the `boot-start` command. If the user that you want to run Cribl Streams does not exist, create it prior to executing. E.g., running Cribl Stream 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 Stream 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 Stream 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 Stream, you can use the following initd commands:

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

Persisting Overrides on initd

Notes on preserving required permissions across restarts and upgrades:

Do NOT Run Cribl Stream as Root!

If Cribl Stream is required to listen on ports 1–1024, it will need privileged access. On a Linux system with POSIX capabilities, you can achieve this by adding the `CAP_NET_BIND_SERVICE` capability. For example: `# setcap cap_net_bind_service=+ep $CRIBL_HOME/bin/cribl`

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

Important: Upgrading Cribl Stream will remove the `CAP_NET_BIND_SERVICE` capability from the `cribl` executable, so you'll need to re-run the appropriate `setcap` command again after each upgrade.

Upon starting the Cribl Stream server, a `bind EACCES 0.0.0.0:<port>` error in the API or Worker logs (depending on the service) might indicate that `setcap` did not successfully execute.

System Proxy Configuration

For details on configuring Cribl Stream to send and receive data through proxy servers, see our System Proxy Configuration topic.

Scaling Up

A single-instance installation can be configured to scale up and utilize as many resources on the host as required. See Sizing and Scaling for details.

Anti-Virus Exceptions

If you are running anti-virus software on a Cribl Stream instance's host OS, here are general guidelines for minimizing accidental blockage of Cribl Stream's normal operation.

Your overall goals are to prevent the anti-virus software from locking any files while Cribl Stream needs to write to them, and from triggering any changes that Cribl Stream 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 Workers or Leader)

Finally, avoid scanning any processes. Except for the queueing/staging directories already listed above, Cribl Stream runs everything in memory, so scanning process memory will slow down Cribl Stream's processing and reduce throughput.

4.2.3. Distributed Deployment

Getting started with a distributed Cribl Stream deployment

To sustain higher incoming data volumes, and/or increased processing, you can scale from a single instance up to a multi-instance, distributed deployment. The Worker instances are centrally managed by a single Leader Node, which is responsible for keeping configurations in sync, and for tracking and monitoring the instances' activity metrics.

- See common distributed deployment use cases in Worker Groups – What Are They and Why You Should Care.
- To sustain high availability, Cribl Stream offers failover to an alternate Leader.
- For detailed system requirements and constraints not listed on this page, see Single-Instance/ Basic Deployment.

As of version 3.0, Cribl Stream's former "master" application components are renamed "leader." While some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect that.

Concepts

Single Instance – a single Cribl Stream instance, running as a standalone (not distributed) installation on one server.

Leader Node – a Cribl Stream instance running in **Leader** mode, used to centrally author configurations and monitor Worker Nodes in a distributed deployment.

Worker Node – a Cribl Stream instance running as a **managed Worker**, whose configuration is fully managed by a Leader Node. (By default, will poll the Leader for configuration changes every 10 seconds.)

Worker Group – a collection of Worker Nodes that share the same configuration. You map Nodes to a Worker Group using a Mapping Ruleset.

Worker Process – a Linux process within a Single Instance, or within Worker Nodes, that handles data inputs, processing, and output. The process count is constrained by the number of physical or virtual CPUs available; for details, see Sizing and Scaling.

Mapping Ruleset – an ordered list of filters, used to map Workers Nodes into Worker Groups.

Options and Constraints

A Worker Node's local running config can be manually overridden/changed, but changes won't persist on the filesystem. To permanently modify a Worker Node's config: Save, commit, and deploy it from the Leader. See Deploying Configurations below.

With an Enterprise license, you can configure Permissions-based (Stream 4.2 and later) or Role-based access control at the Worker Group level. Non-administrator users will then be able to access Workers only within those Worker Groups on which they're authorized.

Aggregating Workers

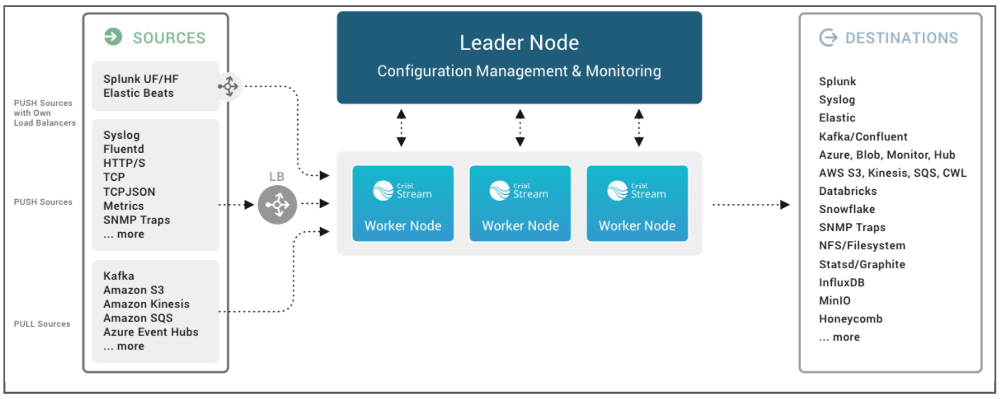
To clarify how the above concepts add up hierarchically, let's use a military metaphor involving toy soldiers:

- Worker Process = soldier.
- Worker Node = multiple Worker Processes = squad.
- Worker Group = multiple Worker Nodes = platoon.

Multiple Worker Groups are very useful in making your configuration reflect organizational or geographic constraints. E.g., you might have a U.S. Worker Group with certain TLS certificates and output settings, versus an APAC Worker Group and an EMEA Worker Group, each with their own distinct certs and settings.

Architecture

This is an overview of a distributed Cribl Stream deployment's components.



Distributed deployment architecture

Here is the division of labor among components of the Leader Node and Worker Node.

Leader Node

- API Process – Handles all the API interactions.
- *N* Config Helpers – One process per Worker Group. Helps with maintaining configs, previews, etc.

Worker Node

- API Process – Handles communication with the Leader Node (i.e., with its API Process) and handles other API requests.
- *N* Worker Processes – Handle all the data processing.

Single-Instance Architecture

For comparison, here's the simpler division of labor on a single-instance deployment, where the separate Leader versus Worker Nodes are essentially condensed into one stack:

- API Process – Handles all the API interactions.
- *N* Worker Processes – Handle all data processing,
- One of the Worker Processes is called the leader Worker Process. (Not to be confused with the Leader Node.) This is responsible for writing configs to disk, in addition to data processing.

So here, the API Process handles the same responsibilities as a Leader Node's API Process, while the Worker Processes correspond to the Worker Nodes' Worker Processes. The exception is that one Worker Process does double duty, also filling in for one of the Leader Node's Config Helpers.

System Requirements

Leader Node Requirements

- **OS**:
  - Linux kernel >= 3.10 and glibc >= 2.17.
    Example distributions: Ubuntu 16.04, Debian 9, RHEL 7, CentOS Linux 7, 8, or CentOS Stream 9, SUSE Linux Enterprise Server 12, Amazon Linux 2.
    Tested so far on Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2.
- **System**:
  - 1 GHz (or faster), 64-bit CPU.
  - 4+ physical cores, 8 GB+ RAM – all beyond your basic OS/VM requirements.
  - 5 GB free disk space.
- **Git**: `git` must be available on the Leader Node. See details below.

- **Browser Support**: The five most-recent versions of Chrome, Firefox, Safari, and Microsoft Edge.

> We assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs). All quantities listed above are minimum requirements.
>
> We recommend deploying the Leader on stable, highly available infrastructure, because of its role in coordinating all Worker instances.

Worker Node Requirements

- See Single-Instance Deployment for requirements. (That page also covers options for enabling a Leader and Workers to automatically start on boot – including how to persist privileged port access on systemd and initd.)
- See Sizing and Scaling for capacity planning details.

Port Requirements

During installation of any Leader or Worker instance, firewalls on that instance's host must enable outbound communication to `https://cdn.cribl.io` on port 443.

Other ports must remain open continuously for a Distributed deployment to function.

If any of this traffic must go through a proxy – except for cluster communication, which can't be proxied – see System Proxy Configuration for configuration details.

FIPS Mode Requirements

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

See the FIPS Mode topic for system and password requirements, and instructions running in FIPS mode.

Installing on Linux

See Single-Instance Deployment, as the installation procedures are identical.

Version Control with `git`

Cribl Stream requires `git` (version 1.8.3.1 or higher) to be available locally on the host where the Leader Node will run. **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 Worker Groups.
- Provide users with an audit trail of all configuration changes.
- Allow users to display diffs between current and previous config versions.

Setting Up Leader and Worker Nodes

This section covers:

1. Configuring a Leader Node
2. Configuring a Worker Node

Configuring a Leader Node

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

Using the UI

At **Settings** > **Distributed Settings** > **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.

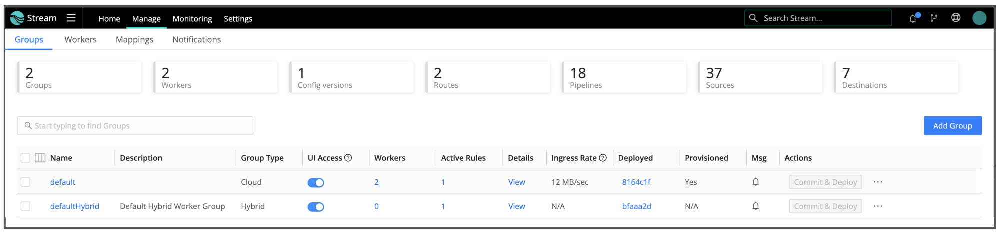
In Cribl Stream 4.0.3 and above, if you later configure Leader High Availability/Failover, further **Distributed Settings** changes will be locked in each Leader's UI. You can still update distributed settings by modifying configuration files, as outlined on the page linked above.

UI Access (Teleporting)

This useful option enables you to click through from the Leader's **Manage Worker Nodes** page to an authenticated view of each Worker's UI. The instructions below correspond to enabling the `groups.yml` file's `workerRemoteAccess` configuration key.

To enable (remote) UI access to Workers from the Leader's UI:

1. From Cribl Stream’s top nav, click **Manage** > **Groups**.
2. On the **Groups** tab: For each desired Worker Group, toggle **UI Access** to `On`.



Name	Description	Group Type	UI Access	Workers	Active Rules	Details	Ingress Rate	Deployed	Provisioned	Msg	Actions
default		Cloud	●	2	1	View	12 MB/sec	8164c1f	Yes		Commit & Deploy
defaultHybrid	Default Hybrid Worker Group	Hybrid	●	0	1	View	N/A	bfaaa2d	N/A		Commit & Deploy

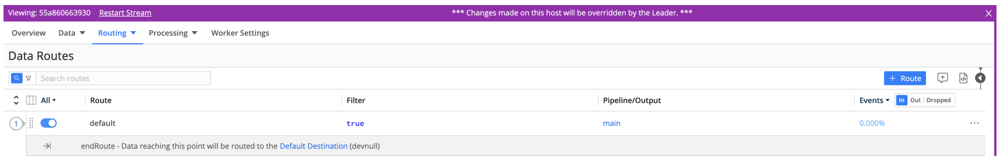
Worker UI access setting

On Cribl.Cloud, the **Groups** page adds extra columns. You’ll see each Group’s Worker type (hybrid versus Cloud/Cribl-managed) and, for Cloud Groups, the anticipated ingress rate and Provisioned status. For details, see Cribl.Cloud Worker Groups.

3. Select the **Workers** tab.
4. On the **Manage Workers** page: Click the link for any Worker you want to inspect.

To confirm that you are remotely viewing a Worker’s UI, Stream will add a magenta border, with a badge labeled **Viewing host:** `<host/GUID>`.

At the upper right, you can click **Restart** to restart the Worker, or click the **X** close button to return to the **Manage Workers** page. Note that any changes you make here will not be propagated to the Leader.



Viewing: 55a860663930 Restart Stream

*** Changes made on this host will be overridden by the Leader. ***

	Route	Filter	Pipeline/Output	Events
1	default	`true`	main	0.000%

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Authenticated view of a Worker

Any changes that you make here will not be propagated to the Leader.

Prior to Cribl Stream 3.4, **Worker UI access** was a global setting per deployment. If you upgrade from a pre-3.4. version to v.3.4 or higher, regardless of your prior configuration, **(Remote) UI access** will initially be set to `No` for all Worker Groups. Re-enable access for each desired Group, as shown above.

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
```

Using the Command Line

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

```
./cribl mode-master [options] [args]
```

For all options, see the CLI Reference.

Auth Token

When you configure a node to work as the Leader, switching from a Single-instance to a Distributed mode, Cribl Stream generates a random auth token.

Configuring a Worker Node

On each Cribl Stream instance you designate as a Worker Node, you can configure the Worker through the UI, the `instance.yml` config file, environment variables, or the command line.

Temporary License Restriction on Worker Node Startup

When you first start a Worker Node, logs might warn about a limit on Worker Processes:

```
{
  "time": "2023-02-10T15:11:48.609Z",
  "cid": "api",
  "channel": "cribl",
  "level": "warn",
  "message": "number of worker processes has been limited by the license",
  "requested": 14,
  "licenseLimit": 10,
  "source": "cribl.log"
}
```

Workers use the default license when they first start up, so a log entry like this one is expected. Once the worker checks with the Leader and gets a production license, the limit on Worker processes is lifted. You can ignore this message.

Using the UI

At **Settings** > **Global Settings** > **Distributed Settings** > **General Settings**, select **Mode**: **Managed Worker (managed by Leader)**.

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.

If you need to customize the Leader's port (from its default `4200`), you can do so via the CLI's mode-worker command.

Using YAML Config File

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

$CRIBL_HOME/local/_system/instance.yml

```
distributed:
  mode: worker
  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 Worker Nodes via environment variables, as in this example:

```
+CRIBL_DIST_MASTER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@masterHostname:4200
./cribl start
```

See the Environment Variables section for more details.

Using the Command Line

You can configure a Worker Node using CLI commands of this form:

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

Cribl Stream will need to restart after this command is issued.

Troubleshooting Resources

Cribl University offers a Troubleshooting Criblet on Attaching Workers to Leaders. 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.

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 Workers 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 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`

Then restart the Cribl server, via the UI or command line.

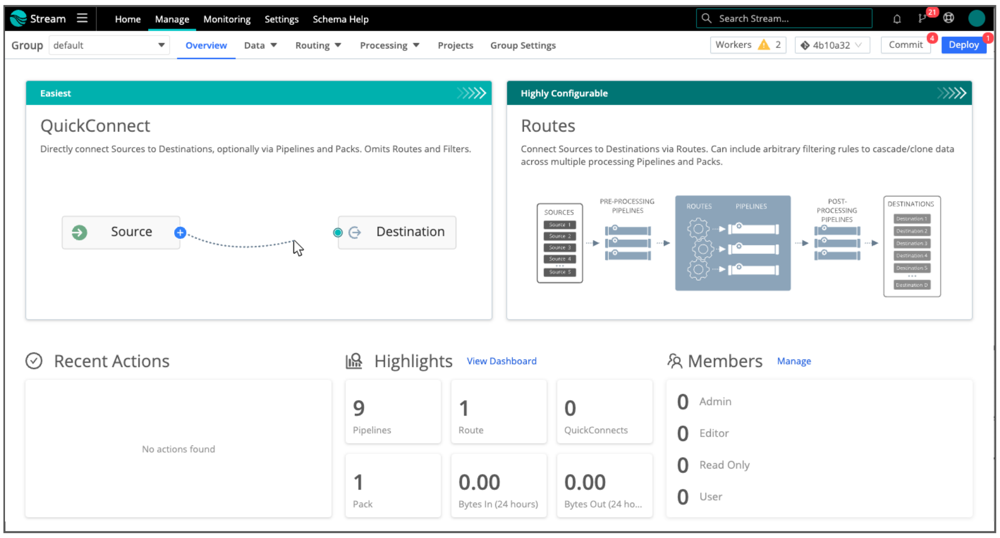
Move the Sockets

Alternatively, you can move Cribl's socket files to a different directory. This directory must be 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**.

Menu Changes in Distributed Mode

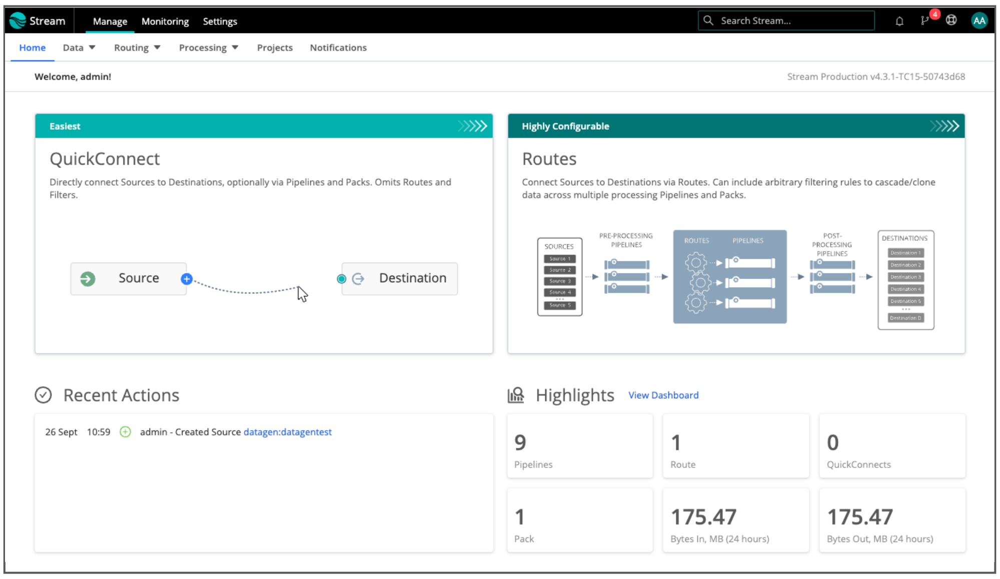
Compared to a single-instance deployment, deploying in distributed mode changes Cribl Stream's menu structure in a few ways, enabling you to manage Workers and their assignments.



Distributed deployment: menu structure

To access the Group-specific top nav shown above, click **Manage**, then click the **Workers** tab. On the **Manage Workers** page click into your desired Worker Group. This contextual top nav also adds a **Settings** tab, through which you can manage configuration per Worker Group.

For comparison, here is a single-instance deployment's Home page:


Stream
Manage
Monitoring
Settings
Search Stream...
Home
Data
Routing
Processing
Projects
Notifications
Welcome, admin!
Stream Production v4.3.1-TC15-50743d68
Easiest
QuickConnect
Directly connect Sources to Destinations, optionally via Pipelines and Packs. Omits Routes and Filters.
Source
Destination
Highly Configurable
Routes
Connect Sources to Destinations via Routes. Can include arbitrary filtering rules to cascade/clone data across multiple processing Pipelines and Packs.
Recent Actions
26 Sept 10:59 admin - Created Source datagen:datagentest
Highlights
View Dashboard
9 Pipelines
1 Route
0 QuickConnects
1 Pack
175.47 Bytes In, MB (24 hours)
175.47 Bytes Out, MB (24 hours)

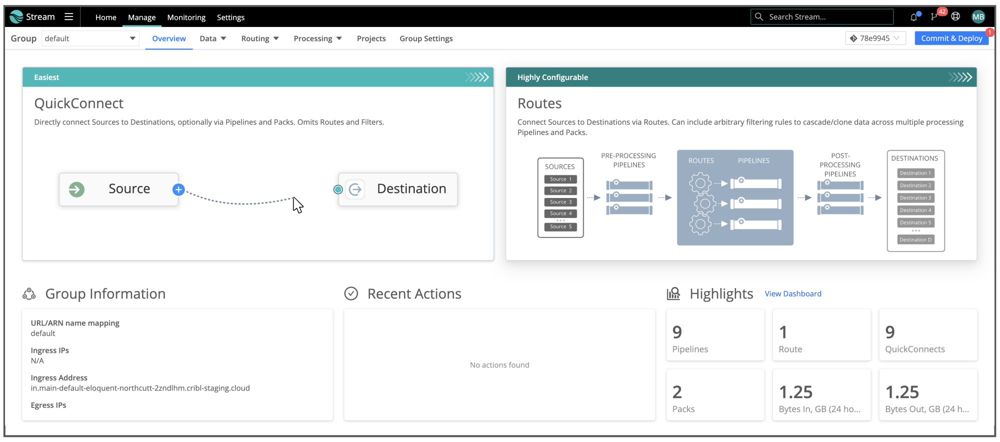
Home page in a single-instance deployment

Commit and Deploy Controls

Distributed mode adds a second set of (Group-specific) **Commit** and **Deploy** buttons at the upper right. The division of labor between these version-control buttons versus the global **Commit** button (on the top nav's **Global Config** dialog) is:

- The upper-right buttons commit and deploy configurations for the currently selected Worker Group/Fleet.
- The top nav's **Commit** button commits configurations for **all** Groups/Fleets, and for the Leader itself – but does not deploy the new configs to Groups/Fleets.

These controls will appear slightly differently depending on whether you have changes committed but not yet deployed, and on whether you've enabled the Collapse actions setting.



Directly connect Sources to Destinations, optionally via Pipelines and Packs. Omits Routes and Filters.

Connect Sources to Destinations via Routes. Can include arbitrary filtering rules to cascade/clone data across multiple processing Pipelines and Packs.

URL/ARN name mapping
default
Ingress IPs
N/A
Ingress Address
in.main-default-eloquent-northcutt-2zndlhm.cribl-staging.cloud
Egress IPs

No actions found

9 Pipelines	1 Route	9 QuickConnects
2 Packs	1.25 Bytes In, GB (24 ho...	1.25 Bytes Out, GB (24 h...

Commit and Deploy controls per Group/Fleet, at upper right

Distributed mode's repositioning of navigation/menu links also applies to several instructions and screenshots that you'll see throughout this documentation.

Where procedures are written around a single-instance scenario, just click into your appropriate Group to access the corresponding navigation links.

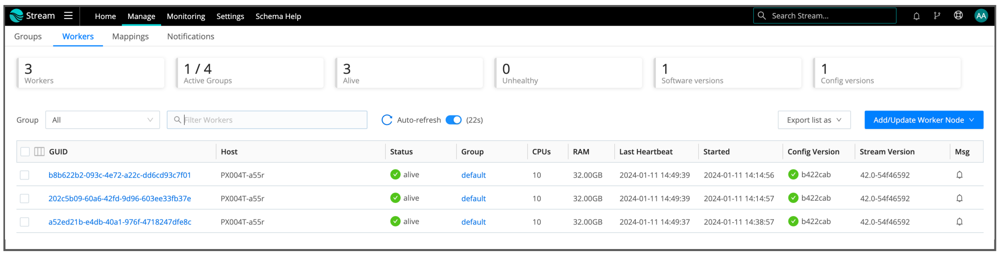
Managing Worker Nodes

To manage Worker Nodes, click the **Manage** tab in the top nav. This opens the Groups landing page with these upper tabs: **Groups**, **Workers**, and **Mappings**. (With a paid license, you will also see a Notifications tab to their right.)

Workers Tab

The **Workers** tab provides status information for each Worker Node in the selected Worker Group. An **Auto-refresh** toggle (on by default) ensures that the **Config Version** icon automatically refreshes when you commit or deploy. Clicking the toggle initiates a refresh of this UI.

This option appears only in Worker Groups that contain fewer than 100 Worker Nodes. It is available in Cribl Stream beginning with v.4.4.4.



GUID	Host	Status	Group	CPUs	RAM	Last Heartbeat	Started	Config Version	Stream Version	Msg
b8b622b2-093c-4e72-a22c-dd6cd93c7f01	PX004T-a55r	✓ alive	default	10	32.00GB	2024-01-11 14:49:39	2024-01-11 14:14:56	✓ b422cab	42.0-54f46592	
202c5b09-60a6-42fd-9d96-603ee33fb37e	PX004T-a55r	✓ alive	default	10	32.00GB	2024-01-11 14:49:39	2024-01-11 14:14:57	✓ b422cab	42.0-54f46592	
a52ed21b-e4db-40a1-976f-4718247dfe8c	PX004T-a55r	✓ alive	default	10	32.00GB	2024-01-11 14:49:37	2024-01-11 14:38:57	✓ b422cab	42.0-54f46592	

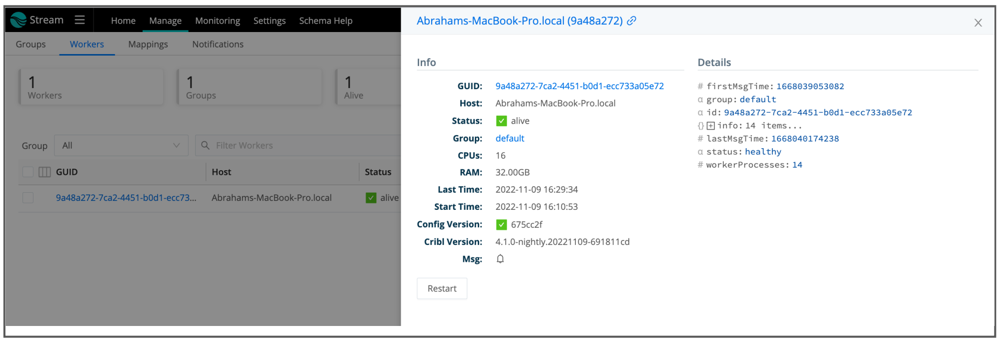
Worker Nodes status/controls

If you see unexpected results here, keep in mind that:

- Workers that miss 5 heartbeats, or whose connections have closed for more than 30 seconds, will be removed from the list.
- For a newly created Worker Group, the **Config Version** column can show an indefinitely spinning progress spinner for that Group's Workers. This happens because the Workers are polling for a config bundle that has not yet been deployed. To resolve this, click the **Deploy** option to force a deploy.

To export the list of Worker Nodes, click the **Export list as** drop-down. Select `JSON` or `CSV` as the export file format. If a list has any filters applied, the exported list will also be filtered. To export an entire list, remove any applied filters.

To reveal more details about a Worker Node, click anywhere on its row.



Abrahams-MacBook-Pro.local (9a48a272)

Info

GUID:	9a48a272-7ca2-4451-b0d1-ecc733a05e72
Host:	Abrahams-MacBook-Pro.local
Status:	✓ alive
Group:	default
CPUs:	16
RAM:	32.00GB
Last Time:	2022-11-09 16:29:34
Start Time:	2022-11-09 16:10:53
Config Version:	✓ 675cc2f
Cribl Version:	4.1.0-nightly.20221109-691811cd
Msg:	

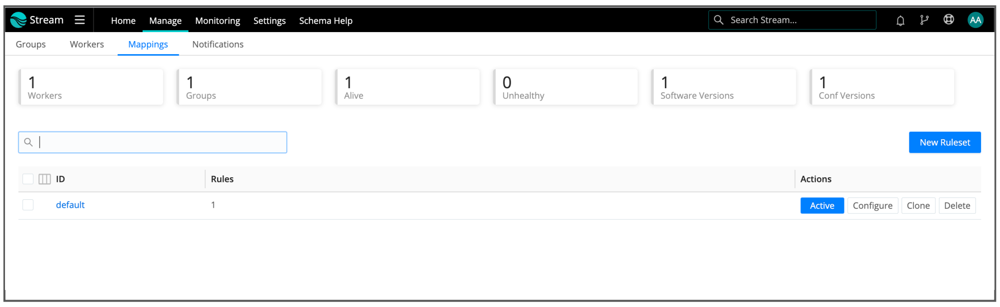
Details

```
# firstMsgTime: 1668039053082
a group: default
a id: 9a48a272-7ca2-4451-b0d1-ecc733a05e72
{} info: 14 items...
# lastMsgTime: 1668040174238
a status: healthy
# workerProcesses: 14
```

Worker Node details

Mappings Tab

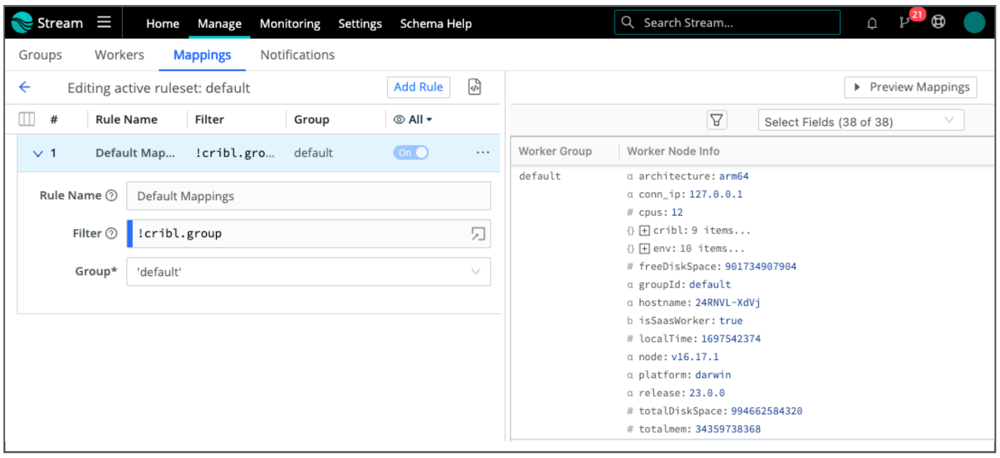
Click the **Mappings** tab to display status and controls for the active Mapping Ruleset:



ID	Rules	Actions
default	1	Active Configure Clone Delete

Mappings status/controls

Click into a Ruleset to manage and preview its contained Rules:



Editing active ruleset: default

#	Rule Name	Filter	Group	All
1	Default Map...	!cribl.gro...	default	On

Rule Name: Default Mappings

Filter: !cribl.group

Group*: 'default'

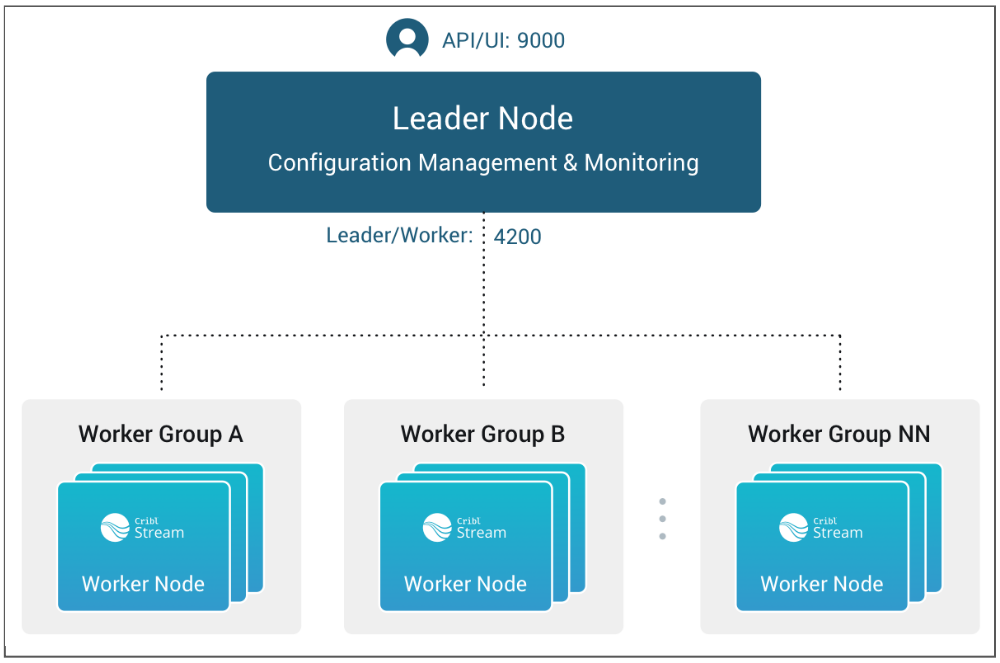
Worker Group	Worker Node Info
default	architecture: arm64 conn_ip: 127.0.0.1 cpus: 12 cribl: 9 items... env: 10 items... freeDiskSpace: 901734907904 groupId: default hostname: 24RNVL-XdVj isSaasWorker: true localTime: 1697542374 node: v16.17.1 platform: darwin release: 23.0.0 totalDiskSpace: 994662584320 totalmem: 34359738368

Editing the Active Ruleset

How Workers and Leader Work Together

The Leader Node has two primary roles:

1. Serves as a central location for Workers' 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 Worker Groups.



Leader Node/Worker Nodes relationship

Network Port Requirements (Defaults)

- UI access to Leader Node: TCP 9000.
- Worker Node to Leader Node: TCP 4200. Used for Heartbeat/Metrics/Leader requests/notifications to clients (for example: live captures, teleporting, status updates, config bundle notifications, and so on).
- Worker Node to Leader Node: HTTPS 4200. Used for config bundle downloads.

Leader/Worker Node Communication

Workers 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.

A Worker Node's failure to successfully send two consecutive heartbeat messages to the Leader will cause the respective Worker to be removed from the Workers page in the Leader's UI, until the Leader again receives a heartbeat message from the affected Worker.

When a Worker Node checks in with the Leader:

- The Worker sends heartbeat to Leader.
- The Leader uses the Worker's configuration, tags (if any), and Mapping Rules to map it to a Worker Group.
- The Worker Node pulls its Group's updated configuration bundle, if necessary.

Safeguarding In-Flight Data When a Worker Node Fails

If a Worker Node goes down, it loses any data that it's actively processing. With streaming senders (Push Sources), Cribl Stream normally handles that data only in-memory. So, to minimize the chance of data loss, configure persistent queues on Sources and/or Destinations that support it.

Leader/Worker Dependency

If the **Leader** goes down, Worker Groups can continue autonomously receiving and processing incoming data from most Sources. They can also continue executing Collection tasks already in process.

However, if the Leader fails, future scheduled Collection jobs will also fail. So will data collection on the Amazon Kinesis Streams, Prometheus Scraper, and all Office 365 Sources – which function as Collectors under the hood.

Config Bundle Management

Config bundles are compressed archives of all config files and associated data that a Worker 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 Worker pulls bundles from the Leader and manages them as follows:

- Last 5 bundles and backup files are kept.
- At any point in time, all files created in the last 10 minutes are kept.
- Bundle cleanup is invoked after a reconfigure.

Bundle in S3

You can store bundles in an AWS S3 bucket to reduce the Leader load. This offloads bundle distribution from the Leader to your designated S3 bucket. Worker and Edge Nodes will pull bundles directly from S3, minimizing Leader strain.

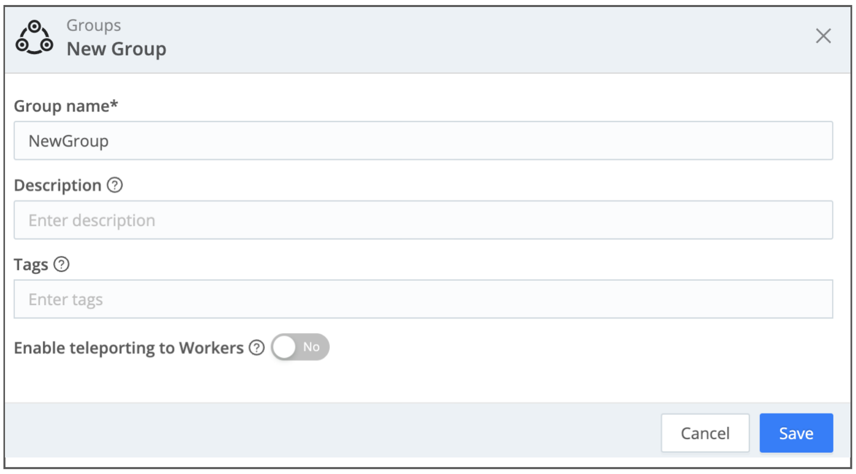
You can configure bundling in S3 through the following ways:

- **UI**: When configuring **Leader Settings** on a Leader Node, specify an S3 bucket (format: `s3://${bucket}`) for remote bundle storage in the **S3 Bundle Bucket URL** field.
- **YAML**: In the `instance.yml` config file, specify the S3 bucket in `master.configBundles.remoteUrl`.
- **Environment Variable**: Use the `CRIBL_DIST_LEADER_BUNDLE_URL` environment variable.

Worker Groups

Worker Groups facilitate authoring and management of configuration settings for a particular set of Workers. To create a new Worker Group, click the top nav's **Manage** tab, then make sure the submenu's **Groups** tab is selected. From the resulting **Groups** tab, click **Add Group**.

For on-prem or Cribl.Cloud hybrid Groups, enter a unique **Name** and (optionally) a **Description** and **Tags**. You can **Enable teleporting to Workers** now or later (details here).


Groups
New Group

Group name*
NewGroup

Description
Enter description

Tags
Enter tags

Enable teleporting to Workers: No

Cancel
Save

Creating a New Group

Configuring multiple Worker Groups, or configuring more than 10 Worker Processes, requires a Cribl Stream Enterprise or Standard license.

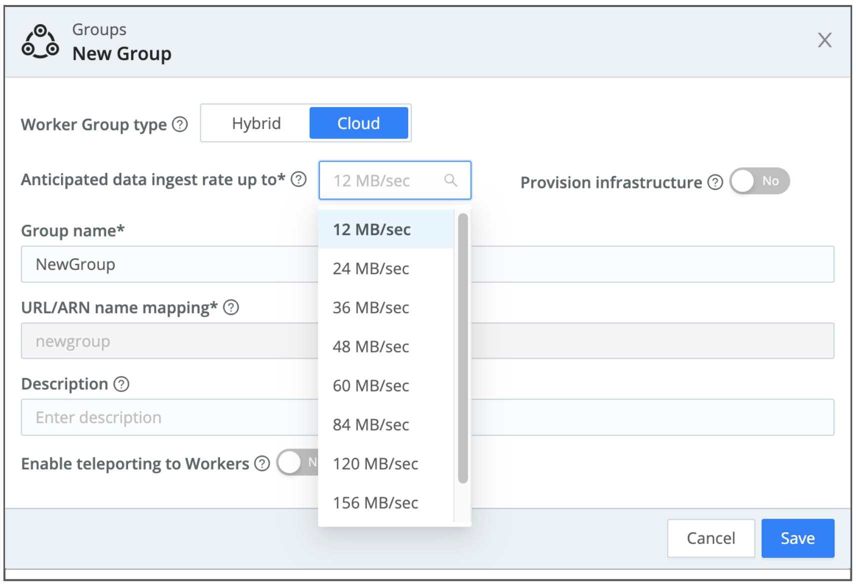
Cribl.Cloud Worker Groups

In Cribl Stream 4.1 and later, Cribl.Cloud supports multiple Groups of Cribl-managed ("Cloud") Workers. (This expands earlier support for multiple Groups of hybrid Workers.) With this feature enabled – which requires an Enterprise plan – Cribl.Cloud's **New Group** modal will provide additional controls.

Your first choice is to select either the **Hybrid** or the **Cloud** button. Selecting **Cloud** adds **Anticipated data ingest rate** and **Provision infrastructure** controls.

Because adding more Groups can require Cribl to provision more infrastructure for you, you must anticipate how many MB/sec of data the new Group will ingest. When you create the Group, you indicate this using the drop-down shown below. Cribl will provision corresponding capacity, assuming a 1:2 ingest:egress ratio.

For general guidance about capacity required for your deployment, see our Planning and Sizing topics.


Groups
New Group

Worker Group type: Hybrid [✓ Cloud]

Anticipated data ingest rate up to*: 12 MB/sec

Provision infrastructure: No

Group name*: NewGroup

URL/ARN name mapping*: newgroup

Description: Enter description

Enable teleporting to Workers: No

Adding and sizing a new Cloud Group

Controlling Costs

Cribl will draw down your Cribl.Cloud credits to cover the base cost of maintaining the infrastructure, as well as your normal cost for ingested data. However, you can manage your costs. One way is to set the **Anticipated data ingest rate** to its minimum `12 MB/sec` value. This will provision the minimum 2 Workers for this Group.

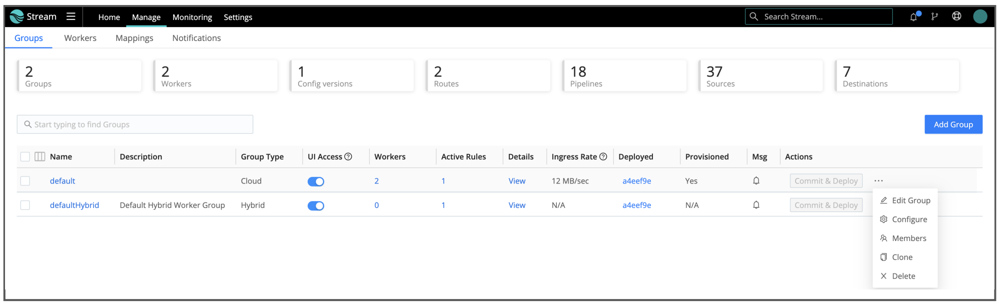
For the greatest savings, you can set **Provision infrastructure** to its `No` default. You can do so either when creating your Group, or at any later time when you are ingesting no data. With this setting, the Group will go dormant, and your base infrastructure cost for this Group will be `0`. In this state, the Group will contain no Workers and will ingest no data. But Stream will retain its configuration, enabling you to easily reprovision it when needed.

Provisioning and Managing Infrastructure

If you set **Provision Infrastructure** to `Yes`, the infrastructure will be provisioned about 30 minutes later. You will receive an in-portal notification confirming the change. The new Group will then process data according to youre new configuration, until you request a new infrastructure change.

You can change all these selections at any later time from the **Manage** > **Groups** page. As shown below, to change a Group's anticipated ingest rate, click the ••• (Options) menu on its row. and select **Edit Group**.

You can also disable the Group here by using the toggle on its row. This will stop data flow, and will stop infrastructure and ingest charges, while preserving the Group's configuration.



Name	Description	Group Type	UI Access	Workers	Active Rules	Details	Ingress Rate	Deployed	Provisioned	Msg	Actions
default		Cloud		2	1	View	12 MB/sec	a4eef9e	Yes		Commit & Deploy
defaultHybrid	Default Hybrid Worker Group	Hybrid		0	1	View	N/A	a4eef9e	N/A		Commit & Deploy

Changing an existing Group's sizing or provisioning

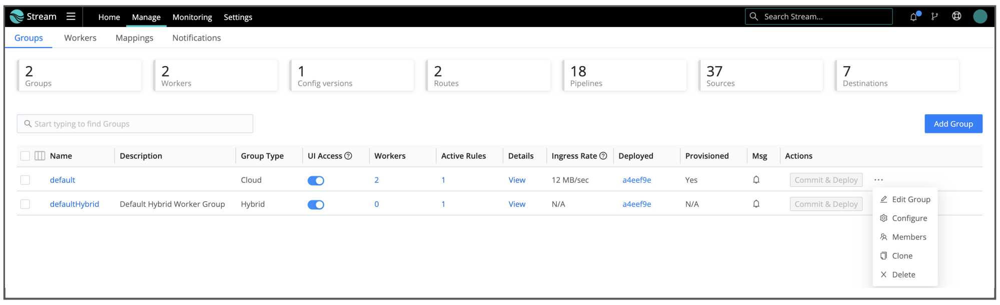
Configuring a Worker Group

To configure a newly created Group, select the top nav's **Manage** tab, then select the **Groups** tab below that, and then click the Group's name. This displays a submenu from which you can configure everything for this Group as if it were a Cribl Stream single instance – using a similar visual interface for Routes, Pipelines, Sources, Destinations, and Group-specific **Settings**.

Managing Worker Groups

On the Groups landing page's right side, beside each Worker Group's **Configure** and **Commit and Deploy** buttons, an ••• (Options) menu provides selections to clone Worker Groups' configurations to new Groups, or to delete Groups.

The `Clone` option copies everything configured on the original Group: Sources, Pipelines, Packs, Routes, and Destinations.



Name	Description	Group Type	UI Access	Workers	Active Rules	Details	Ingress Rate	Deployed	Provisioned	Msg	Actions
default		Cloud		2	1	View	12 MB/sec	a4eef9e	Yes		Commit & Deploy
defaultHybrid	Default Hybrid Worker Group	Hybrid		0	1	View	N/A	a4eef9e	N/A		Commit & Deploy

Manage Groups page: Options menu

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

To explicitly set passwords for Worker Groups, see User Authentication.

Recovering a Worker Group

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

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

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

Creating a New Worker Group with the Same Name

1. Create a new Worker Group with the same name as the one that was deleted. This action will restore the deleted Worker Group to the system.
2. Select a previous version of the deleted Worker Group from the Version Control menu if you previously committed the deletion of your Worker Group.

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

Temporarily Stopping the Cribl Server

1. Stop the Cribl server.
2. Identify the last commit before the Worker Group was deleted.
3. Revert to that commit.
4. Restart the Cribl server.

Mapping Workers to Worker Groups

Mapping Rulesets are used to map Workers to Worker Groups. Within a ruleset, a list of rules evaluate Filter expressions on the information that Workers 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 Worker Group should be defined and present in the system.

The ruleset behavior is similar to Routes, where the order matters, and the **Filter** section supports full JavaScript expressions. The ruleset matching strategy is first-match, and one Worker can belong to only one Worker Group.

Creating a Mapping Ruleset

To create a Mapping Ruleset, click the top nav's **Manage** tab, then click the submenu's **Mappings** tab. Click **Add Ruleset**, assign a unique **ID** in the resulting **New Ruleset** modal, and click **Save**.

The **Mappings** link appears only when you have started Cribl Stream with **Settings** > **Global Settings** > **Distributed Settings** > **Mode** set to **Leader**.

On the resulting **Manage Mapping Rulesets** page, 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 Worker Groups.

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, Workers can be configured to send a Group with their payload. See below how this ranks in mapping priority.

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 `w0`, AND:
- Belongs to `Group420`.

Rule Configuration

- **Rule Name**: `myFirstRule`
- **Filter**: `(conn_ip.startsWith('10.10.42.') && cpus > 6) || env.CRIBL_HOME.match('w0')`
- **Group**: `Group420`

Default Worker Group and Mapping

When a Cribl Stream instance runs as Leader, the following are created automatically:

- A `default` Worker Group.
- A `default` Mapping Ruleset,
    - with a `default` Rule matching all (`true`).

Mapping Order of Priority

Priority for mapping to a group is as follows: Mapping Rules > Group sent by Worker > `default` Group.

- If a Filter matches, use that Group.
- Else, if a Worker has a Group defined, use that.
- Else, map to the `default` Group.

Deploying Configurations

Your typical workflow for deploying Cribl Stream 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 Workers.** You deploy new configurations at the Group level: Locate your desired Group and click on **Deploy**. Workers that belong to the group will start **pulling** updated configurations on their next check-in with the Leader.

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

When a Worker 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 Worker Node with default credentials. For details, see User Authentication.

Configuration Files

On the Leader, a Worker Group's configuration lives under:
`$CRIBL_HOME/groups/<groupName>/local/cribl/`.

On the managed Worker, after configs have been pulled, they're extracted under:
`$CRIBL_HOME/local/cribl/`.

Lookup Files

On the Leader, a Group's lookup files live under: `$CRIBL_HOME/groups/<groupName>/data/lookups`.

On the managed Worker, after configs have been pulled, lookups are extracted under: `$CRIBL_HOME/data/lookups`. When deployed via the Leader, lookup files are distributed to Workers as part of a configuration deployment.

If you want your lookup files to be part of the Cribl Stream configuration's version control process, we recommended deploying using the Leader Node. Otherwise, you can update your lookup file out-of-band on the individual Workers. The latter is especially useful for larger lookup files (> 10 MB, for example), or for lookup files maintained using some other mechanism, or for lookup files that are updated frequently.

For other options, see Managing Large Lookups.

Some configuration changes will require restarts, while many others require only reloads. See here for details.

Restarts/reloads of each Worker Process are handled automatically by the Worker. Note that individual Worker Nodes might temporarily disappear from the Leader's **Workers** tab while restarting.

Worker Process Rolling Restart

During a restart, to minimize ingestion disruption and increase availability of network ports, Worker Processes on a Worker Node are restarted in a rolling fashion. **20% of running processes – with a minimum of one process – are restarted at a time.** A Worker Process must come up and report as **started** before the next one is restarted. This rolling restart continues until all processes have restarted. If a Worker Process fails to restart, configurations will be rolled back.

Auto-Scaling Workers and Load-Balancing Incoming Data

If data flows in via Load Balancers, make sure to register all instances. Each Cribl Stream Node exposes a health endpoint that your Load Balancer can check to make a data/connection routing decision.

Health Check Endpoint	Healthy Response
`curl http://<host>:<port>/api/v1/health`	`{"status":"healthy"}`

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`
  - `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 Worker 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`.

When you include an auth token in `CRIBL_DIST_MASTER_URL` to deploy managed Cribl Stream or Edge Nodes, it can include only alphanumeric characters or the underscore (`_`). Do not include `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`.

- `CRIBL_DIST_MODE` – `worker | 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_DIST_LEADER_BUNDLE_URL` – AWS S3 bucket (format: `s3://${bucket}`) for remote bundle storage.
- `CRIBL_TMP_DIR` – Defines the root of a temporary directory.
  Sources use this 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_VOLUME_DIR` – Sets a directory that persists modified data between different containers or ephemeral instances. When set, `CRIBL_VOLUME_DIR` 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. 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`.

See GitOps for a separate list of GitOps-oriented environment variables.

Deprecated Variables

These were removed as of LogStream 3.0:

- `CRIBL_CONFIG_LOCATION`.
- `CRIBL_SCRIPTS_LOCATION`.

Workers GUID

When you install and first run the software, Cribl Stream generates a GUID which it stores in a `.dat` file located in `CRIBL_HOME/local/cribl/auth`, e.g.:

```
$ cat /opt/cribl/local/cribl/auth/676f6174733432.dat
{"it":1647910738,"phf":0,"guid":"e0e48340-f961-4f50-956a-5153224b34e3","lics":["li
```

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

Host Info/Metadata

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

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

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

You can customize the type of metadata collected at **Group Settings** > **Limits** > **Metadata**. Use the **Event metadata sources** drop-down list (and/or the **Add source** button) to add and select metadata sources.

In Cribl Stream mode, all the **Event metadata sources** are disabled by default.

The 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 Stream version, distributed mode, Workers (for managed instances), and config version.
- `aws`: Reports details on 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 Stream 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. And `kube` applies only to on-prem deployments.

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

`__metadata.kube` Property

For the `__metadata.kube` property (`kube` on the UI) to report details on a Kubernetes environment, Cribl Stream needs to figure out where it is running. Set the `KUBE_K8S_POD` environment variable to the name of the Pod in which Cribl Stream 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 Stream 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.

4.2.4. Splunk App Deployment

Getting started with Cribl App for Splunk

Deploying Cribl App for Splunk

In an on-prem Splunk environment, you can install and configure Cribl Stream as a Splunk app (Cribl App for Splunk), on a Search Head. You **cannot** use Cribl App for Splunk in a Cribl Stream distributed deployment as a Leader, or as a managed Worker. You **cannot** install the app on Splunk Cloud or on a Splunk Heavy Forwarder. (See Single-Instance/Basic Deployment and Distributed Deployment for alternatives.)

Running on a Search Head (SH)

When running on an SH, Cribl Stream is set to **mode-searchhead**, the default mode for the app. It listens for **localhost traffic** generated by a custom command: `criblstream`. The command is used to forward search results to the Cribl Stream instance's TCP JSON input on port `10420`, but it's also capable of sending to any other Cribl Stream instance listening for TCP JSON.

Once received, data can be processed and forwarded to any of the supported Destinations. In addition, several out-of-the box saved searches are ready to run and send their results to Cribl with a single click.

Installing the Cribl App for Splunk on an SH

- Select an instance on which to install.
- Ensure that ports `10000`, `10420`, and `9000` are available. See the Requirements section for more info.
- Get the bits here, and install as a regular Splunk app.
- Restart the Splunk instance.
- Go to `https://<instance>/en-US/app/cribl` or `https://<instance>:9000`, and log in with Splunk **admin** role credentials.

Typical Use Cases for Search Head Mode

- Working with search results in a Cribl Stream pipeline.
- Sending search results to any Destination supported by Cribl Stream.

Commands and Examples

Below are some examples of how to use the `criblstream`, `criblencrypt`, and `cribldecrypt` commands available in this app.

The `cribldecrypt` command can decrypt events originating from multiple Stream Worker Groups or Edge Fleets **only** if you use the same key material on all Worker Groups and Fleets encrypting data.

Forward Search Results to a Cribl Stream Instance

The basic command for sending events to a Cribl Pipeline is `criblstream`, for example:

criblstream command

```
index="my_index" | criblstream dest=host:port
```

Search for a Value

Suppose that you want to search events for a particular value. Because events are encrypted, one approach would be to decrypt all events and then search the unencrypted events for your target value. But decrypting large numbers of events takes a long time, and is extremely inefficient.

Here's a better approach, taking advantage of the fact that the encrypted form of a value is always the same:

1. Encrypt the target value.
2. Search the events (which are still encrypted) for the encrypted form of the target value.
3. If you find events that contain the (encrypted) target value, decrypt them if desired.

You can combine all three steps in a single command:

Search the index for the encrypted target value and decrypt matching events

```
index="encrypted" \
[ | makeresults | eval encrypted_field="target_value" \
| criblencrypt field=encrypted_field keyclass= 0 \
| fields encrypted_field ] | cribldecrypt
```

The `criblencrypt` command requires a `field`, and either a `keyclass` (as shown above) or a `keyid`. For details, see Encryption.

Internal Fields

The Cribl App for Splunk adds the following internal fields to events:

- `_CRIBL_QUEUE`
- `_CRIBL_TCP_ROUTING`

The app forwards these fields to Splunk, because the `transforms.conf` file that ships with the app relies on them.

Beginning with version 4.0, only the Cribl App for Splunk adds these fields. Cribl Stream and Cribl Edge do not. (Earlier versions of Cribl Stream did add these fields, but neither Stream itself nor Splunk receivers have processed them in recent versions.)

4.2.5. Bootstrap Workers From Leader

Boot fully provisioned Workers

Cribl Stream Workers can completely provision themselves, directly from the Leader, upon initial boot. This means that a group of any number of Workers can launch and be fully functional within the cluster, in seconds.

How Does It Work?

A Cribl Stream Leader Node provides a bootstrap API endpoint, at `/init/install-worker.sh`, which returns a shell script. You can run this shell script on any supported machine (see Restrictions below), without Cribl Stream installed. This fully provisions the machine as a Worker Node.

Although you can specify the download URL when you execute the initial curl command, the Cribl Stream package is not downloaded until you generate the script via the API, and then execute it.

For additional information on using environment variables to set the hostname, see Setting the Hostname.

Requirements

All Worker Nodes' hosts must enable ongoing outbound communication to the Leader's port 4200, to enable the Leader to manage the Workers. While the bootstrap script runs, firewalls on each Worker's host must also allow outbound communication on the following ports:

- Port 443 to `https://cdn.cribl.io`.
- Port 443 to a Cribl.Cloud Leader.
- Port 9000 to an on-premises Leader.

If any of this traffic must go through a proxy, see System Proxy Configuration for configuration details. To anticipate and resolve edge cases not mentioned here, see Restrictions below.

Troubleshooting Root Access or SSL Errors

The script will install Cribl Stream into `/opt/cribl`, and will make system-level changes. For systems like Ubuntu, which don't allow direct root access, you'll need to use the `sudo` prefix when executing the script.

> The script will create a user named `cribl` to install, own, and run Cribl Stream/Edge.
>
> If you encounter errors of this form:
> `ssl certificate problem: self signed certificate in certificate chain`
> …add the `-k` flag to disable certificate validation.

UI Access

Cribl Stream 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 a Worker's Group assignment.

> To use these options, you must have the `admin` Role on a distributed deployment's Leader Node, with an Enterprise license. You must also create the Worker Groups before using the following instructions to add or reassign Workers to them.

Add New Worker

1. From Cribl Stream's top nav, select **Manage** > **Groups**.
2. Click the link in your desired Group's **Workers** column.
3. On the resulting **Workers** tab, click **Add/Update Worker Node** at the upper right.
4. Select **Linux** > **Add** from the menu, as shown in the composite screenshot below.
5. In the resulting **Add Linux Worker** modal, the **Install package location** defaults to `Cribl CDN`.
   If desired, change this to `Download URL`. (For details about this option, see Adding Download URL.)
6. As needed, correct the target **Group**, **User**, and **User Group**, as well as the **Leader hostname/IP** (URI).
7. Copy the resulting script to your clipboard.
8. Click **Done** to close the modal.
9. Paste the script onto your Worker Node's command line and execute it, to add the Worker.
   - As needed (see the note above), prepend `sudo` to the generated **Script** field's contents, and/or append the `-k` flag.

Update Existing Worker

You can also auto-generate a script that will update an existing Worker's Group assignment, and/or assign the Worker to a different Leader:

1. From Cribl Stream's top nav, select **Manage** > **Groups**.
2. Click the link in your desired Group's **Workers** column.

3. From the **Add/Update Worker Node** menu, select **Linux** > **Update**.
4. In the resulting **Update Configuration on Linux Workers** modal, select the new target **Group** for this Worker.
5. As needed, correct or change the **Leader hostname/IP** and/or **Leader port number**.
6. The **Script type** drop-down defaults to the `Environment variable` option for generating the script's command. If you're not relying on environment variables, change this to `CLI`.
7. Copy the resulting script to your clipboard.
8. Click **Done** to close the modal.
9. Paste the script onto your Worker Node's command line and execute it, to update the Worker's assignment.
    - As needed (see the note above), prepend `sudo` to the generated **Script** field's contents, and/or append the `-k` flag.

API Spec

Request Format

```
GET http://<leader hostname or IP>:9000/init/install-worker.sh
```

Query Strings

String	Required?	Description
`token`	optional	Leader Node's shared secret (`authToken`). By default, this is set to a random string. You can find this secret in the Leader Node's **Distributed Settings** section. Before v4.5.0, the default token was `criblmaster`. This token is still in use in instances upgraded from an earlier version.
`group`	optional	Name of the cluster's Worker Group. If not specified, falls back to `default`.
`download_url`	optional	Provide the complete URL to a Cribl Stream installation binary. This is especially useful if the Worker Nodes don't have access to the Internet to download from cribl.io.
`tag`	optional	When used in conjunction with Mapping Rules, enables you to specify the Worker Group you want the bootstrapped Worker to join. Multiple tags should be in the form `&tag=tag1&tag=tag2`.
`user`	optional	The user to run Cribl Stream. Defatuls to `cribl`.

String	Required?	Description
`user_group`	optional	The user group that owns all files. Defaults to `cribl`.
`install_dir`	optional	The file path to install Cribl Stream. Defaults to `/opt/cribl`.

Example HTTP Request

```
GET http://<leader hostname or IP>:9000/init/install-worker.sh?token=79364d6e-dead-
```

As of version 3.0, Cribl Stream's former "master" application components are renamed "leader." While some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect that.

Example Response

```
#!/bin/sh

### START CRIBL LEADER TEMPLATE SETTINGS ###

CRIBL_MASTER_HOST="<Master FQDN/IP>"
CRIBL_AUTH_TOKEN="<Auth token string>"
CRIBL_VERSION="<Version>"
CRIBL_GROUP="<Default group preference>"
CRIBL_MASTER_PORT="<Master heartbeat port>"
CRIBL_DOWNLOAD_URL="<download url>"

### END CRIBL MASTER TEMPLATE SETTINGS ###

# Set defaults
checkrun() { $1 --help >/dev/null 2>/dev/null; }
faildep() { [ $? -eq 127 ] && echo "$1 not found" && exit 1; }
[ -z "${CRIBL_MASTER_HOST}" ] && echo "CRIBL_MASTER_HOST not set" && exit 1
CRIBL_INSTALL_DIR="${CRIBL_INSTALL_DIR:-/opt/cribl}"
CRIBL_MASTER_PORT="${CRIBL_MASTER_PORT:-4200}"
CRIBL_AUTH_TOKEN="${CRIBL_AUTH_TOKEN:-criblmaster}"
CRIBL_GROUP="${CRIBL_GROUP:-default}"
if [ -z "${CRIBL_DOWNLOAD_URL}" ]; then
    FILE="cribl-${CRIBL_VERSION}-linux-x64.tgz"
    CRIBL_DOWNLOAD_URL="https://cdn.cribl.io/dl/$(echo ${CRIBL_VERSION} | cut -d '
fi
UBUNTU=0
CENTOS=0
AMAZON=0

echo "Checking dependencies"
checkrun curl && faildep curl
checkrun adduser && faildep adduser
checkrun usermod && faildep usermod
BOOTSTART=1
SYSTEMCTL=1
checkrun systemctl && [ $? -eq 127 ] && BOOTSTART=0
checkrun update-rc.d && [ $? -eq 127 ] && BOOTSTART=0

echo "Checking OS version"
lsb_release -d 2>/dev/null | grep -i ubuntu && [ $? -eq  0 ] && UBUNTU=1
cat /etc/system-release 2>/dev/null | grep -i amazon && [ $? -eq 0 ] && AMAZON=1
```

```
echo "Creating cribl user"
if [ $UBUNTU -eq 1 ]; then
    adduser cribl --home /home/cribl --gecos "Cribl Stream User" --disabled-passwo
fi
if  [ $CENTOS -eq 1 ] || [ $AMAZON -eq 1 ]; then
    adduser cribl -d /home/cribl -c "Cribl Stream User" -m
    usermod -aG wheel cribl
fi

echo "Installing Cribl Stream"
mkdir -p ${CRIBL_INSTALL_DIR}
curl -Lso ./cribl.tar.gz "${CRIBL_DOWNLOAD_URL}"
tar xzf ./cribl.tar.gz -C ${CRIBL_INSTALL_DIR} --strip-components=1
rm -f ./cribl.tar.gz
chown -R cribl:cribl ${CRIBL_INSTALL_DIR}

if [ $BOOTSTART -eq 1 ]; then
    echo "Setting Cribl Stream to start on boot"
    ${CRIBL_INSTALL_DIR}/bin/cribl boot-start enable -u cribl
fi

mkdir -p ${CRIBL_INSTALL_DIR}/local/_system
cat <<-EOF > ${CRIBL_INSTALL_DIR}/local/_system/instance.yml
distributed:
  mode: worker
  master:
    host: ${CRIBL_MASTER_HOST}
    port: ${CRIBL_MASTER_PORT}
    authToken: ${CRIBL_AUTH_TOKEN}
    tls:
      disabled: true
  group: ${CRIBL_GROUP}
EOF

chown -R cribl:cribl ${CRIBL_INSTALL_DIR}
if [ $BOOTSTART -eq 1 ]; then
  service cribl start
else
  ${CRIBL_INSTALL_DIR}/bin/cribl start
fi
```

curl Option

An easy way of wrapping HTTP methods is to use the `curl` command. Here is an example, which uses a `GET` operation by default, with the same URL used in the above HTTP example:

```
curl http://<leader hostname or IP>:9000/init/install-worker.sh?token=79364d6e-dea
```

Check Requirements above to avoid/resolve port or ownership issues.

Chaining Script Execution

The `GET` and `curl` procedures above will only output the contents of the script that needs executing – the script will still need to be manually executed.

However, you can automate that part, too, using a command like the one shown below. This passes the script's contents to the `bash` shell to immediately execute.

```
curl http://<leader hostname or IP>:9000/init/install-worker.sh?token=79364d6e-dea
```

As noted above, on Ubuntu and similar systems, you might need to insert `sudo` before the `bash`. If you don't have a bash shell available, you can pipe the command to `| sh -` instead.

Adding Download URL

By default, the script gets configured to download the Cribl Stream package from the public Cribl repository. If you want to specify a different download location in the script, you'll use the `download_url` parameter.

To successfully execute the `curl` command while also specifying the download URL, you must enclose the URL in double quotes. The reason for this is that the `&` character that joins multiple HTTP parameters is interpreted by the shell as the operator to run commands in the background. Double-quoting the URL, as shown in this example, prevents this.

```
curl "http://<leader hostname or IP>:9000/init/install-worker.sh?token=79364d6e-de
```

curl Offline Option

To bootstrap Workers in an internet-disconnected (e.g., airgapped) environment, you can separate downloading the installation package from Cribl's repository versus running the `curl` command. Here's an example:

1. Download the installation package from Cribl's download page.
2. Rename the resulting file from `cribl-x.x.x-xxxxxxxx-linux-x64.tgz` to `cribl.tar.gz`.
3. Bootstrap the Worker from the file's current directory, by running a command of this form:

```
curl "https://<mycriblleader.mydomain.ext>:9000/init/install-worker.sh?group=defaul
```

Tagging to Assign Workers to Worker Groups

Cribl Stream uses Mapping Rulesets to map Workers to Worker Groups. When you create a Worker from a bootstrap script, you can take advantage of Mapping Rulesets to specify which Worker Group you want the newly-created Worker to join. This is done by adding tags to the download URL, in the form `&tag=tag1&tag=tag2`.

Basic Example

Suppose you have a Worker Group, `Group420`, that you want bootstrapped Workers to join.

In the active Mapping Ruleset for `Group420`, create a new rule that maps any Worker with the tag `awseast1` to the `Group420` Worker Group. You can do this with a filter:

```
cribl.tags.include('awseast1')
```

Then, in the bootstrap script, add a download URL with a tag that matches the filter you just created. For example:

```
curl "http://<logstream_leader>:9000/init/install-worker.sh?tag=awseast1&token=<tol
```

When you use the script to bootstrap a new Worker, the Worker will be assigned to `Group420`.

Advanced Example

Suppose you have four Worker Groups distributed between two regions and two platforms:

	AWS	Azure
Region 1	`Group01`	`Group02`
Region 2	`Group03`	`Group04`

Using two tags (one for region and one for platform) you can represent all four possible combinations, and thus all four Worker Groups:

- `tag=aws&tag=region1` maps to `Group01`.
- `tag=azure&tag=region1` maps to `Group02`.
- `tag=aws&tag=region2` maps to `Group03`.
- `tag=azure&tag=region2` maps to `Group04`.

For each Worker Group, you'll create a Mapping Rule with a filter for the appropriate tag combination to match. For example, this filter would match `Group04`:

```
cribl.tags.includes('azure') && cribl.tags.includes('region2')
```

Then you can use the tag combination in the download URL of a bootstrap script. For example, the tags in this download URL map to `Group04`:

```
http://<logstream_leader>:9000/init/install-worker.sh?tag=azure&tag=region2&token=<
```

Any Worker created by the script with the above download URL will be assigned to `Group04`.

Status Codes

Status Code	Reason
200 – OK	All is well. You should have received the script as a response.
403 – Forbidden	Either the node is not configured as a Leader, or the token provided is invalid.

Restrictions

Keep the following in mind when using bootstrap scripts:

- Each Worker must normally have access to the internet in order to download the Cribl Stream installation binary from cribl.io. Where this isn't feasible, you can use the `download_url` switch to point to a binary in a restricted location.
- TLS is not enabled by default. If enabled and configured, access to this feature will be over `https` instead of `http`.
- Red Hat, Ubuntu, CentOS, and Amazon Linux are the only supported Worker platforms.

User Data

For public-cloud customers, an easy way to use bootstrap scripts is in an instance's user data. First, be sure to set the Leader Node to `mode = 'leader'`. Then use the following script (changing the command as needed. based on the information above). Upon launch, the Worker Node will reach out to the Leader, download the script, download the Cribl Stream package from the specified location, and then install and configure Cribl Stream:

```
#!/bin/bash
curl http://<leader-node-ip/host-address>:9000/init/install-worker.sh?token=<auth-t
```

4.2.6. Leader High Availability/Failover

To handle unexpected outages in your on-premises distributed deployment, Cribl Stream 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.

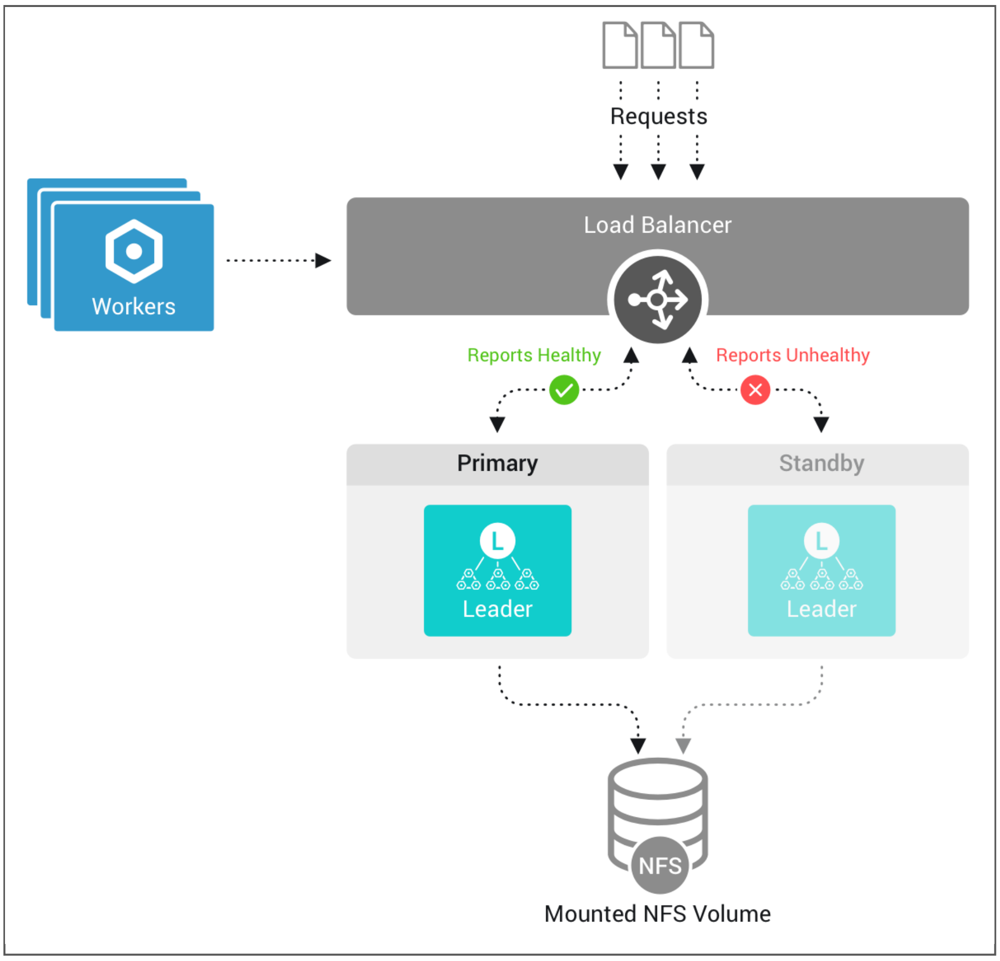
Configuring a backup Leader requires a Cribl Stream Enterprise or Standard license.

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 the same Network File System (NFS) volume/shared drive.

If the primary Leader Node goes down:

- Cribl Stream 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 Worker 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, Worker 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 Stream Worker 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 Stream. To do the same for folders, add the `reldiratime` option.
2. `noatime`: Never update the access time. (Cribl Stream 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). Suitable for TCP, UDP, and TLS traffic.
- AWS Application Load Balancer (ALB). Application-aware, suitable for HTTP/HTTPS traffic.
- HAProxy.
- NGINX Plus.

AWS Network Load Balancers

If you need to access the same target through a Network Load Balancer, use an IP-based target group and deactivate client IP preservation. For details, see:

- Why can an instance in a target group not reach itself via NLB?
- Why can't a target behind my Network Load Balancer connect to its own Network Load Balancer?

Recommended Configuration

Use the latest NFS client across all Leaders. If you are on AWS, we recommend the following:

- Use Amazon's Elastic File System (AWS EFS) for your NFS storage.
- Ensure that the user running Cribl Stream 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

Remember, the `$CRIBL_VOLUME_DIR` environment variable overrides `$CRIBL_HOME`.

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 Stream 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

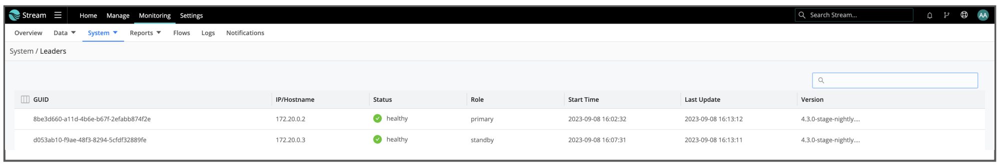
You can configure additional Leader Nodes via the following environment variables:

- `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**.



GUID	IP/Hostname	Status	Role	Start Time	Last Update	Version
8be3d660-a11d-4b6e-b67f-2efabb874f2e	172.20.0.2	✓ healthy	primary	2023-09-08 16:02:32	2023-09-08 16:13:12	4.3.0-stage-nightly....
d053ab10-f9ae-48f3-8294-5cfdf32889fe	172.20.0.3	✓ healthy	standby	2023-09-08 16:07:31	2023-09-08 16:13:11	4.3.0-stage-nightly....

Monitoring Leader Nodes

Upgrading

When upgrading:

1. Stop both Leaders.
2. Upgrade (Stream, Edge) the primary Leader.
3. Upgrade the second Leader.
4. Start both Leaders again, one by one.
5. Upgrade each Worker 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.

4.2.7. Converting A Single Instance To Distributed Deployment

If you've configured a Cribl Stream single-instance deployment and now want to promote it to distributed, here are a couple of approaches to doing so, while retaining the configuration you've already created.

Simple Copy

We'll start with the simplest scenario, in which you plan to set up distributed mode with a single Worker Group. By default, this group will literally be named `default`. (We'll also explain how to extend this scenario.)

1. If you haven't already installed `git` (required for the Leader), do so as outlined here.
2. Stop the Cribl Stream server (`./cribl stop`).
3. Your single instance's configs are under Cribl Stream's `local/` subdirectory. So, copy `$CRIBL_HOME/local/cribl/*` to `$CRIBL_HOME/groups/default/local/cribl/`. This stages your configs for the `default` Worker Group.
4. Restart Cribl Stream, selecting **Distributed Mode**: `Leader`.
5. At this point, the Leader should have inherited your previous single-instance settings. Commit and deploy these settings to the `default` group, which should resume the same data processing that your single instance was executing.
6. If you want to replicate the same configs to additional Worker Groups, add those groups now via the **Manage** > **Groups** UI. Then repeat the preceding four steps, targeting the new subdirectories that have been created on the filesystem for the new groups.

Creating multiple Worker Groups requires an Enterprise or Standard license.

rsync

This alternative approach uses `rsync` to replicate your single-instance configs.

1. Use this command to rsync your single-instance configuration to each of your distributed Groups (replacing the `<group-name>` placeholder here):

```
rsync -a $cribl/local/cribl newmaster:$cribl/groups/<group-name>/local/
```

2. Restart Cribl Stream, selecting **Distributed Mode**: `Leader`.
3. Commit and deploy the new configuration.

4.3. Orchestrated Deployment

4.3.1. Kubernetes/Helm Deployment

Boot fully provisioned Leader and Worker Nodes via Helm

Cribl's leader and workergroup Helm charts provide a fast way to deploy a distributed Cribl Stream environment to a Kubernetes cluster.

For up-to-date details on Prerequisites, Deploying, Upgrading, and running distributed on a Free License, see the Cribl Helm Charts repo's Readme.

K8s Leader Deployment

The rest of this page outlines how to deploy a Cribl Stream Leader Node (or single instance) to AWS via Kubernetes, using a Cribl-provided Leader Helm chart.

This chart is a work in progress, provided as-is. Cribl expects to further develop and refine it.

Cribl recommends deploying the Leader on stable, highly available infrastructure, because of its role in coordinating all Worker instances.

Deprecation Notice

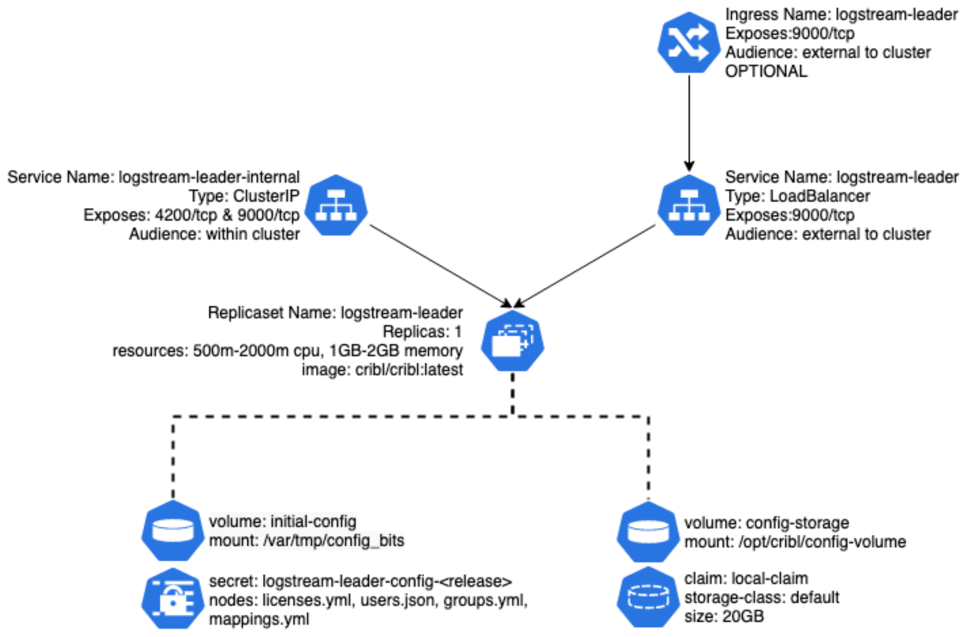
This chart replaces the `logstream-master` chart, which was deprecated as of v.2.9.9. See Migration below for instructions on migrating to this new chart to access features newly provided in this and future versions.

New Capabilities

- Supports Cribl Stream v.4.3.1.

Deployment

As built, Cribl's chart will deploy a Cribl Stream Leader server for Cribl Stream, consisting of a deployment, two services, and a number of persistent volumes.


Ingress Name: logstream-leader
Exposes:9000/tcp
Audience: external to cluster
OPTIONAL
Service Name: logstream-leader-internal
Type: ClusterIP
Exposes: 4200/tcp & 9000/tcp
Audience: within cluster
Service Name: logstream-leader
Type: LoadBalancer
Exposes:9000/tcp
Audience: external to cluster
Replicaset Name: logstream-leader
Replicas: 1
resources: 500m-2000m cpu, 1GB-2GB memory
image: cribl/cribl:latest
volume: initial-config
mount: /var/tmp/config_bits
secret: logstream-leader-config-<release>
nodes: licenses.yml, users.json, groups.yml, mappings.yml
volume: config-storage
mount: /opt/cribl/config-volume
claim: local-claim
storage-class: default
size: 20GB

Deployment schematic

Note that this chart creates two load-balanced services:

- The main one (named after the Helm release), which is intended as the primary service interface for users.
- The "internal" one (named `<helm-release>-internal`), which is intended for the worker-group-to-leader communication.

By default, this chart installs only a Cribl Stream Leader Node. To also deploy Cribl Stream Worker Groups/Fleets via Helm, you can use the Set Up Worker Groups/Mappings override described below.

You can also use Cribl's separate logstream-workergroup chart. For details, see Kubernetes Deployment: Worker Group/Fleet in this documentation.

AWS and Kubernetes Prerequisites

This section covers both general and specific prerequisites, with a bias toward the EKS-oriented approach that Cribl uses for its own deployments.

Set Up AWS CLI

Install the AWS CLI, version 2, following AWS' instructions.

Next, create or modify your `~/.aws/config` file to include (at least) a `[profile]` section with the following SSO (single-sign-on) details:

&#126;/.aws/config

```
[profile <your-profile-name>]
sso_start_url = https://<your-domain>/start#/
sso_region = <your-AWS-SSO-region>
sso_account_id = <your-AWS-SSO-account-ID>
sso_role_name = <your-AWS-role-name>
region = <your-AWS-deployment-region>
```

Set Up kubectl

You will, of course, need `kubectl` set up on your local machine or VM. Follow Kubernetes' installation instructions.

Add a Cluster to Your kubeconfig File

You must modify your `~/.kube/config` file to instruct kubectl what cluster (context) to work with.

1. Run a command of this form: `aws --profile <profile-name> eks update-kubeconfig --name <cluster-name>`
   This should return a response like this: `Added new context arn:aws:eks:us-west-2:424242424242:cluster/<cluster-name> to /Users/<username>/.kube/config`
2. In the resulting `~/.kube/config` file's `args` section, as the new first child, insert the profile argument that you provided to the aws command. For example:

&#126;/.kube/config

```
args:
- --profile=<profile-name>
- --region
[...]
```

3. Also change the `command: aws` pair to include the full path to the `aws` executable. This is usually in `/usr/local/bin`, in which case you'd insert: `command: /usr/local/bin/aws`.

This section of `~/.kube/config` should now look something like this:

```
      args:
      - --profile=<profile-name>
      - --region
      - us-west-2
      - eks
      - get-token
      - --cluster-name
      - lab
      command: /usr/local/bin/aws
      env:
      - name: AWS_PROFILE
        value: <profile-name>
```

With these AWS and Kubernetes prerequisites completed, you're now set up to run `kubectl` commands against your cluster, as long as you have an active aws SSO login session.

Next, do the Helm setup.

Install Helm and Cribl Repo

1. You'll need Helm (preferably v.3.x) installed. Follow the instructions here.
2. Add Cribl's repo to Helm, using this command: `helm repo add cribl https://criblio.github.io/helm-charts/`

Persistent Storage

The chart requires persistent storage. It will use your default StorageClass, or (if you prefer) you can override `config.scName` with the name of a specific StorageClass to use.

Cribl has tested this chart primarily using AWS EBS storage, via the CSI EBS driver. The volumes are created as `ReadWriteOnce` claims. For details about storage classes, see Kubernetes' Storage Classes documentation.

AWS-Specific Notes

If you're running on EKS, Cribl highly recommends that you use Availability Zone–specific node groups. For details, see eksctl.io's Autoscaling documentation.

Do not allow a single node group to spans AZs. This can lead to trouble in mounting volumes, because EBS volumes are AZ-specific.

See other EKS-Specific Issues on our GitHub repo.

Configure the Chart's Values

You'll want to override some of the chart's default values. The easiest way is to copy this chart's default `values.yaml` file from our repo. save it locally, modify it, and install it in Helm:

1. Copy the **raw** contents of: https://github.com/criblio/helm-charts/blob/master/helm-chart-sources/logstream-leader/values.yaml
2. Save this as a local file, e.g.: `/bar/values.yaml`
3. Modify values as necessary (see Values to Override in the Helm Charts docs).
4. Install your updated values to Helm, using this command: `helm install -f /bar/values.yaml`

Match Versions

Cribl recommends that you use the same Cribl Stream version on Worker Nodes/Edge Nodes versus the Leader Node. So if, for any reason, you're not yet upgrading your Workers to the version in the Leader's default `values.yaml` > `criblImage.tag`, be sure to override that `criblImage.tag` value to match the version you're running on all Workers.

EKS-Specific Values

If you're deploying to EKS, many annotations are available for the load balancer. Set these as values for the `service.annotations` key. Internally, we typically use the annotations for logging to S3, like this:

values.yaml [excerpt]

```
    service.beta.kubernetes.io/aws-load-balancer-access-log-enabled: "true"
    service.beta.kubernetes.io/aws-load-balancer-access-log-emit-interval: "5"
    service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-name: "<buck
    service.beta.kubernetes.io/aws-load-balancer-access-log-s3-bucket-prefix: "ELB
```

For an exhaustive list of annotations you can use with AWS's Elastic Load Balancers, see the Kubernetes Service documentation.

Basic Chart Installation

With the above prerequisites and configuration completed, you're ready to install our chart to deploy a Cribl Stream Leader Node. Here are some example commands:

- To install the chart with the release name `logstream-leader`:
  `helm install logstream-leader cribl/logstream-leader`
- To install the chart using the storage class `ebs-sc`:
  `helm install logstream-leader cribl/logstream-leader --set config.scName='ebs-sc'`

Post-Install/Post-Upgrade

Cribl Stream will not automatically deploy changes to the Worker Nodes/Edge Nodes. You'll need to commit and deploy changes to all of your Worker Groups/Fleets.

Change the Configuration

If you don't override its default values, this Helm chart effectively creates a single-instance deployment of Cribl Stream, using the standard container image. You can later configure distributed mode, licensing, user passwords, etc., all from the Cribl Stream UI. However, you also have the option to change these configuration details upfront, by installing with value overrides. Here are some common examples.

Apply a License

If you have a Standard or Enterprise license, you can use the `config.license` parameter to add it as an override to your install:

```
helm install logstream-leader cribl/logstream-leader --set config.license="<long
encoded license string redacted>"
```

Run Distributed on a Free License

If you do not specify a license with `config.license`, and you want to run distributed, you'll need to go to Cribl Stream's **Settings** > **Global Settings** > **Licensing** UI page and accept the Free license. (The Free license allows one Worker Group/Fleet.)

If your Helm configuration includes the `config.groups` option, the Cribl Stream Leader Node will be configured as a distributed Leader. If you omit that option, it will be configured as a single instance. (You can later use Cribl Stream's **Settings** > **Global Settings** > **Distributed** page to select **Mode:** `Leader`.)

Set the Admin Password

Normally, when you first install Cribl Stream and log into the UI, it prompts you to change the default admin password. You can skip the password-change challenge by setting your admin password via the `config.adminPassword` parameter:

```
helm install logstream-leader cribl/logstream-leader --set config.adminPassword="
<new password>"
```

Set Up Worker Groups/Mappings

As mentioned above, the chart's default is to install a vanilla deployment of Cribl Stream. If you are deploying as a Leader, you can use the `config.groups` parameter to define the Worker Groups/Fleets you want created and mapped. Each group in the list you provide will be created as a Worker Group/Fleet, with a Mapping Rule to seek a tag with that Worker Group's name in it:

```
helm install logstream-leader cribl/logstream-leader --set config.groups=
{group1,group2,group3}
```

The example above will create three Worker Groups/Fleets/Fleets – `group1`, `group2`, and `group3` – and a Mapping Rule for each.

Migrating from the logstream-master Chart

Here is how to migrate from the deprecated `logstream-master` chart to `logstream-leader`.

Exporting your Configuration

You'll need to "export" your data from the existing `logstream-master` pod. And first, you'll need to get the current pod's name, as well as its namespace. The easiest way to do this is to run `kubectl get pods -A` and then look pods that start with the release name you used when you ran `helm install`. For example, if you installed with the following command:

```
helm install ls-master cribl/logstream-master
```

...you'd look for a pod name that started with `ls-master`.

Once you've identified your pod and namespace, you can export your configuration using a combination of `kubectl` and `tar`:

```
kubectl exec <pod name> -n <namespace> -- bash -c "cd /opt/cribl/config-volume; ta
```

This command executes the tar based back up of the config-volume, and outputs it to a local tar file (`cribl_backup.tar`).

“Re-Hydrating” the Backup on the logstream-leader Chart

Exploding the tarball onto the new persistent volume is a one-time event. Once the config volume is restored, you'll make changes to the config via the Cribl Stream UI or API. Either approach will change the config on disk, which you wouldn't want to overwrite the next time the pod restarts. You can manually re-hydrate the backup by installing the logstream-leader chart, and then running the following command:

```
cat cribl_backup.tar| kubectl -n <namespace> exec --stdin <pod name> -- bash -c "c
```

This will restore the data into the config volume (which is mounted as `/opt/cribl/config-volume`). If you want to double-check that, run:

```
kubectl -n <namespace> exec <pod name> -- bash -c "ls -alR /opt/cribl/config-volum
```

After this, you want to **delete** the active pod, allowing the new one to come up with the restored configuration. To do this, you'd run the following `kubectl` command:

```
kubectl -n <namespace> delete <pod name>
```

This will cause the pod to exit, but the deployment will replace it with a new pod which will use the same config persistent volume.

Reconfiguring the Worker Groups/Fleets

Now that you've got a new working leader chart, you need to tell the workers to connect to the new leader instead of to the old `logstream-master` instance. This is a simple `helm upgrade` operation. You'll need to use the same command string that you used to install, changing the word `install` to `upgrade`. But change the value of `config.host` to the new service that was created for the logstream-leader install. (You can change `config.host` either via the `--set` option or in the `values.yml` file.) For example, if you ran the `logstream-leader` install with the release name `ls-lead`, like this:

```
helm install ls-lead -f <values file> cribl/logstream-leader
```

...you'd run `kubectl get service -n <namespace> | grep ls-lead` to get the two services that it created, and you'll want the name of the one that ends in `-internal`. In this case, that name would be `ls-lead-leader-internal`.

Assume that for your workergroup install, you used a release name of `ls-wg1`, and a values file named `my-values.yml` with the following contents:

```
config:
  host: logstream-master-internal
  group: kubernetes
  token: <token>
  rejectSelfSignedCerts: 0
```

...then you'd replace the `host` value in this file with `ls-lead-leader-internal`, and then run:

```
helm upgrade ls-wg1 -f my-values.yml -n <namespace>
```

The upgrade **should** replace all the existing workergroup pods with newly reconfigured ones. However, if you notice any workergroup pods with an AGE value indicating that it was started before the upgrade command, simply kill those pods, and they will re-spawn with the new configuration.

Preloading Configuration

The `extraConfigmapMounts` and `extraSecretMounts` options enable you to preload configuration files into the leader chart, via ConfigMaps and Secrets that you've created in your Kubernetes environment. However, because ConfigMaps and Secret mounts are read-only, you can't simply mount them into the configuration tree.

Therefore, you must mount them to a location outside of the `/opt/cribl` tree, and then copy the files into the tree at startup. This copying can be accomplished using environment variables, as we'll see below.

Both ConfigMaps and Secret mounts can be made writable, but the K8s documentation recommends against this.

Configuration Locations

The chart creates a single configuration volume claim, `config-storage`, which gets mounted as `/opt/cribl/config-volume`. All Worker Group/Fleet configuration lives under the `groups/` subdirectory. If you have a Worker Group/Fleet named `datacenter_a`, its configuration will live in

/opt/cribl/config-volume/groups/datacenter_a. See Configuration Files section for details on file locations.

Using Environment Variables to Copy Files

The cribl container's entrypoint.sh file looks for up to 30 environment variables assumed to be shell-script snippets to execute before Cribl Stream startup (CRIBL_BEFORE_START_CMD_[1-30]). It also looks for up to 30 environment variables to execute after Cribl Stream startup (CRIBL_AFTER_START_CMD_[1-30]).

The variables in each set need to be in order, and cannot skip a number. (The entrypoint.sh script breaks the loop the first time it doesn't find an env var, so if you have CRIBL_BEFORE_START_CMD_1 skipping to CRIBL_BEFORE_START_CMD_3, then CRIBL_BEFORE_START_CMD_3 will not be executed.)

The chart uses this capability to inject the license and to set up groups. We'll use this same capability to copy our config files into place. So if you've provided the config.license and config.groups variables (occupying the first two slots), you'll need to start with CRIBL_BEFORE_START_CMD_3. In the examples below, we'll start with CRIBL_BEFORE_START_CMD_3, assuming that a config.license and config.groups have been set.

Figuring Out Which Variable to Use

The easiest way to figure out which environment variable you need to use is to deploy the chart with all the options you plan to use (i.e., to use the helm install command with options that you plan to use for your deployment). Then check the pod definition for CRIBL_* environment variables. For example, if you used the following install command:

```
% helm install lsms -f ../leader-values.yaml -n logstream-ht cribl/logstream-leade
```

You can now get the pod's name:

```
% kubectl get pods -n logstream-ht
NAME                                READY   STATUS    RESTARTS   AGE
lsms-leader-659bfccdd6-xsz67        1/1     Running   0          52m
```

And then you can use kubectl describe to get the relevant environment variables:

```
% kubectl describe  pod/lsms-leader-659bfccdd6-xsz67 -n logstream-ht  | egrep "CRIE
CRIBL_BEFORE_START_CMD_1:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/licenses.yml
CRIBL_BEFORE_START_CMD_2:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/mappings.yml
CRIBL_AFTER_START_CMD_1:       [ ! -f $CRIBL_VOLUME_DIR/users_imported ] && sleep 2
```

From that, you can tell that we already have a `CRIBL_BEFORE_START_CMD_1` and `CRIBL_BEFORE_START_CMD_2`, so our next logical variable should be `CRIBL_BEFORE_START_CMD_3`.

Preloading Scenario

Here's a preload scenario that includes a sample ConfigMap, `extraConfigmapMounts`, copy command, and copy-once flag.

The ConfigMap

Let's say we want to preconfigure a collector job in the `group1` Worker Group. The job will be called `InfrastructureLogs`, and it will read ELB logs from an S3 bucket. First, we'll need a `jobs.yml` file, like this:

```
  InfrastructureLogs:
    type: collection
    ttl: 4h
    removeFields: []
    resumeOnBoot: false
    schedule: {}
    collector:
      conf:
        signatureVersion: v4
        enableAssumeRole: true
        recurse: true
        maxBatchSize: 10
        bucket: <my infrastructure logs bucket>
        path: /ELB/AWSLogs/${aws_acct_id}/elasticloadbalancing/${aws_region}/${_time
        region: us-west-2
        assumeRoleArn: arn:aws:iam::<accountid>:role/LogReadAssume
      destructive: false
      type: s3
    input:
      type: collection
      staleChannelFlushMs: 10000
      sendToRoutes: false
      preprocess:
        disabled: true
      throttleRatePerSec: "0"
      breakerRulesets:
        - AWS Ruleset
      pipeline: devnull
      output: devnull
```

We'll need this loaded into a ConfigMap object, so we'd run kubectl to create a ConfigMap from the directory where our `jobs.yml` file resides:

```
kubectl create configmap job-config --from-file <containing directory> -n
<deployment namespace>
```

So if that file is in a directory called `./config-dir`, and we're deploying the leader chart into the `logstream` namespace, we'd create it like this:

```
kubectl create configmap job-config --from-file ./config-dir -n logstream
```

extraConfigmapMounts Config

In our `values.yaml` file, we need to specify the ConfigMap and where to mount it:

```
extraConfigmapMounts:
  - name: job-config
    configMap: job-config
    mountPath: /var/tmp/job-config
```

This example will mount the files in the ConfigMap into the pod's `/var/tmp/job-config` directory.

Copying the Config Files

You could simply define, in the `values.yaml` file (or via `--set`):

```
env:
  CRIBL_BEFORE_START_CMD_3: "cp /var/tmp/job-config /opt/cribl/config-volume/groups
```

However, there are two potential problems with that:

1. There is no guarantee that the destination directory tree will be there. (The first time a pod spins up, it won't be.)
2. If the pod has crashed and spun up anew, blindly copying will overwrite any changes previously made. This is rarely desirable behavior.

File Copying Pattern

Since we might want to copy multiple configuration files in one shot, it makes sense to use some sort of "flag file" to ensure that we copy the files only once. The script snippet to copy the `jobs.yaml` file looks like this, formatted for readability:

```
FLAG_FILE=/opt/cribl/config-volume/job-flag
if [ ! -e $FLAG_FILE ]; then
  mkdir -p /opt/cribl/config-volume/groups/group1/local/cribl # ensure the directo
  cp /var/tmp/job-config/jobs.yml /opt/cribl/config-volume/groups/group1/local/crib
  touch $FLAG_FILE
fi
```

This looks to see if the file `/opt/cribl/config-volume/job-flag` exists, and if it doesn't, creates the directory tree, copies the config file(s), and then creates the job flag file. However, we need to format it a little differently to easily encompass it in the `env` variable:

```
  env:
    CRIBL_BEFORE_START_CMD_3: "FLAG_FILE=/opt/cribl/config-volume/job-flag; if [ ! -
```

Once you run `helm install` with this in the `values.yaml` file, you can do `kubectl exec` on the pod to execute a shell:

```
kubectl exec -it <pod name> -- bash
```

...and then look at `/opt/cribl/config-volume/groups/group1/local/cribl/jobs.yml` to verify that it is in place.

Uninstall the Infrastructure

To spin down deployed pods, use the helm uninstall command – where `<release-name>` is the namespace you assigned when you installed the chart:

```
helm uninstall <release-name>
```

You can append the `--dry-run` flag to verify which releases will be uninstalled before actually uninstalling them:

```
helm uninstall <release-name> --dry-run
```

Known Issues

- Cribl's current architecture supports **only** TCP ports in Worker Groups'/Fleets' `service` > `ports` configuration. This restriction might be removed in future versions.
- The upgrade process from pre-2.4.0 versions creates an `initContainer`, which will run prior to any instance of the Cribl Stream pod. Because the coalescence operation will not overwrite existing data, this is not a functional problem. But depending on your persistent-volume setup, the `initContainer`'s precedence might cause pod restarts to take additional time while waiting for the volume claims to release. The only upgrade path that will have this issue is 2.3.* -> 2.4.0. In the next iteration, we'll remove the `initContainer` from the upgrade path.
- The upgrade process leaves the old `PersistentVolumes` and `PersistentVolumeClaims` around. This is, unfortunately, necessary for this upgrade path. In follow-on versions, we will remove these volumes from the chart.
- See EKS-specific issues on our GitHub repo.

4.3.2. Kubernetes Worker Deployment

Boot a fully provisioned Worker Group via Helm.

This page outlines how to deploy a Cribl Stream Worker Group to AWS via Kubernetes, using a Cribl-provided Helm chart.

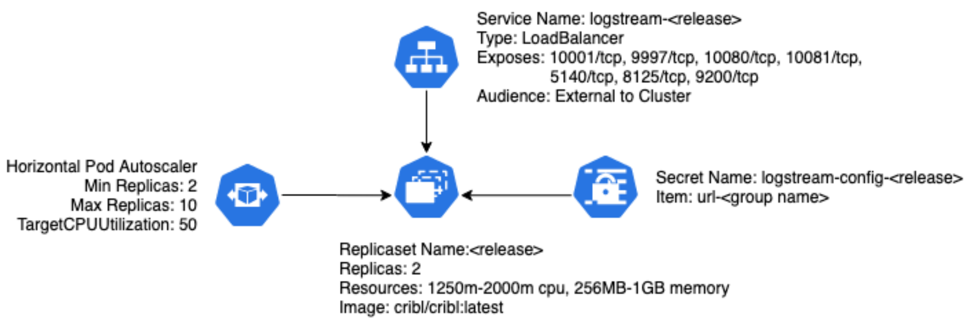
> This chart will deploy only a Cribl Stream Worker Group, whose functioning depends on the presence of a Cribl Stream Leader Node. To deploy the Leader, see Kubernetes Leader Deployment.

New Capabilities

- Supports Cribl Stream v.4.3.1.

Deployment

As built, Cribl's chart will deploy a simple Worker Group for Cribl Stream, consisting of a deployment, a service, a horizontal pod autoscaler configuration, and a secret used for configuration.


Service Name: logstream-<release>
Type: LoadBalancer
Exposes: 10001/tcp, 9997/tcp, 10080/tcp, 10081/tcp, 5140/tcp, 8125/tcp, 9200/tcp
Audience: External to Cluster
Horizontal Pod Autoscaler
Min Replicas: 2
Max Replicas: 10
TargetCPUUtilization: 50
Secret Name: logstream-config-<release>
Item: url-<group name>
Replicaset Name:<release>
Replicas: 2
Resources: 1250m-2000m cpu, 256MB-1GB memory
Image: cribl/cribl:latest

Deployment schematic

AWS and Kubernetes Prerequisites

This section covers both general and specific prerequisites, with a bias toward the EKS-oriented approach that Cribl uses for its own deployments.

Set Up AWS CLI

Install the AWS CLI, version 2, following AWS' instructions.

Next, create or modify your `~/.aws/config` file to include (at least) a `[profile]` section with the following SSO (single-sign-on) details:

&#126;/.aws/config

```
[profile <your-profile-name>]
sso_start_url = https://<your-domain>/start#/
sso_region = <your-AWS-SSO-region>
sso_account_id = <your-AWS-SSO-account-ID>
sso_role_name = <your-AWS-role-name>
region = <your-AWS-deployment-region>
```

Set Up `kubectl`

You will, of course, need `kubectl` set up on your local machine or VM. Follow Kubernetes' installation instructions.

Add a Cluster to Your `kubeconfig` File

You must modify your `~/.kube/config` file to instruct `kubectl` what cluster (context) to work with.

1. Run a command using this format: `aws --profile <profile-name> eks update-kubeconfig --name <cluster-name>`
   This should return a response like this: `Added new context arn:aws:eks:us-west-2:424242424242:cluster/<cluster-name> to /Users/<username>/.kube/config`
2. In the resulting `~/.kube/config` file's `args` section, as the new first child, insert the profile argument that you provided to the `aws` command. For example:

/.kube/config

```
args:
- --profile=<profile-name>
- --region
[...]
```

3. Also change the `command: aws` pair to include the full path to the `aws` executable. This is usually in `/usr/local/bin`, in which case you'd insert: `command: /usr/local/bin/aws`.

This section of `~/.kube/config` should now look something like this:

```
args:
- --profile=<profile-name>
- --region
- us-west-2
- eks
- get-token
- --cluster-name
- lab
command: /usr/local/bin/aws
env:
- name: AWS_PROFILE
  value: <profile-name>
```

With these AWS and Kubernetes prerequisites completed, you're now set up to run `kubectl` commands against your cluster, as long as you have an active aws SSO login session.

Next, install Helm and the Cribl repository.

Install Helm and Cribl Repo

1. You'll need Helm (preferably v.3.x) installed. Follow the instructions here.
2. Add Cribl's repo to Helm, using this command: `helm repo add cribl https://criblio.github.io/helm-charts/`
3. Display the default values available to configure Cribl's `logstream-workergroup` chart: `helm show values cribl/logstream-workergroup`

Configure the Chart's Values

You'll want to override some of the values you've just displayed. The easiest way is to copy this chart's default `values.yaml` file from our repo. save it locally, modify it, and install it in Helm:

1. Copy the **raw** contents of: https://github.com/criblio/helm-charts/blob/master/helm-chart-sources/logstream-workergroup/values.yaml
2. Save this as a local file, e.g.: `/foo/values.yaml`
3. Modify values as necessary (see Values to Override in the Cribl Helm Charts docs). To see the full scope of values available, run: `helm show values cribl/logstream-workergroup`.

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

Match Versions

Cribl recommends that you use the same Cribl Stream version on Leader Nodes versus Worker Group/Fleet Nodes. So, if you're not yet upgrading your Leader to the version in the current `values.yaml` > `criblImage.tag`, be sure to override that `criblImage.tag` value to match the version you're running on the Leader.

Set Worker Processes Statically

For all container-based Worker Groups, Cribl recommends that you directly specify the Worker **Process count** using a positive integer (like `+3`). This convention prevents the overprovisioning of Worker Processes that can occur with dynamic values (like the default `-2` setting).

To access **Process count** in Stream's UI, select **Group Settings** > **System** > **Worker Processes** (distributed deployments) or **Settings** > **System** > **Service Processes** (single-instance deployments).

Set a Maximum Number of Workers

To provision a Kubernetes container with a specific number of Workers, use the `CRIBL_MAX_WORKERS` environment variable. Setting a maximum amount of Workers on a Kubernetes node helps to prevent overloading the node with more CPUs than it can handle.

See Sizing and Scaling for more information on this environment variable and example configurations.

Install the Chart

With the above prerequisites and configuration completed, you're ready to install our chart to deploy a Cribl Stream Worker Group/Fleet. Here are some example commands:

- To install the chart with the release name `logstream-wg`:
  `helm install logstream-wg cribl/logstream-workergroup`
- To install the chart using the Cribl Stream Leader `logstream.lab.cribl.io`:
  `helm install logstream-wg cribl/logstream-workergroup --set config.host='logstream.lab.cribl.io`
- To install the chart using the Cribl Stream Leader `logstream.lab.cribl.io` in the namespace `cribl-helm`:
  `helm install logstream-wg cribl/logstream-workergroup --set config.host='logstream.lab.cribl.io' -n cribl-helm`

Upgrading

You upgrade using the `helm upgrade` command. But it's important to ensure that your Helm repository cache is up to date, so first issue this command:

```
helm repo update
```

After this step, invoke:

```
helm upgrade <release> -n <namespace> cribl/logstream-workergroup
```

For the example above, where the release is `logstream-wg` and is installed in the `cribl-helm` namespace, the command would be:

```
helm upgrade logstream-wg -n cribl-helm cribl/logstream-workergroup
```

This Helm chart's upgrade is idempotent, so you can use the upgrade mechanism to upgrade the chart, but you can also use it to change its configuration (as outlined in Change the Configuration).

Optional: Kubernetes API Access

Versions 2.4.0+ include access mechanisms for Worker Groups to access the Kubernetes API.
The `values.yaml` file provides three relevant options:

- `rbac.create` – Enables the creation of a Service Account, Cluster Role, and Role Binding (which binds the first two together) for the release.
- `rbac.resources` – Specifies the Kubernetes API resources that will be available to the release.
- `rbac.verbs` – Specifies the API verbs that will be available to the release.
- `rbac.extraRules` – Additional rulesets for the cluster role.

For more information on the verbs and resources available, see Kubernetes' Using RBAC Authorization documentation.

Change the Configuration

Once you've installed a release, you can get its `values.yaml` file by using the `helm get values` command. For example, assuming a release name of `logstream-wg`, you could use this command:

```
helm get values logstream-wg -o yaml > values.yaml
```

This will retrieve a local `values.yaml` file containing the values in the running release, including any values that you overrode when you installed the release.

You can now make changes to this local `values.yaml` file, and then use the `helm upgrade` operation to "upgrade" the release with the new configuration.

For example, assume you wanted to add an additional TCP-based syslog port, listening on port 5141, to the existing `logstream-wg` release. In the `values.yaml` file's `service` > `ports` section, you'd add the three key-value pairs shown below:

values.yaml (excerpt)

```
service:
  [...]

  ports:
  [...]
  - name: syslog
    port: 5141
    protocol: TCP
```

Then you'd run:

```
helm upgrade logstream-wg cribl/logstream-workergroup -f values.yaml
```

Remember, if you installed in a namespace, you need to include the `-n <namespace>` option to any `helm` command. You'll still have to create the source in your Cribl Stream Leader, and commit and deploy it to your Worker Group/Fleet.

Using Persistent Storage for Persistent Queueing

The `extraVolumeMounts` option makes it feasible to use persistent volumes for Cribl Stream persistent queueing. However, Cribl does not recommend this combination – there is variability in persistent-storage implementations, and this variability can lead to problems in scaling Worker Groups/Fleets. However, if you choose to implement persistent volumes for queueing, please consider these suggestions:

1. Use a shared-storage-volume mechanism. We've worked with the EFS CSI driver for AWS, and it works fairly well (although it can be tedious to configure).

2. Understand your Kubernetes networking topology, and how that topology interacts with your persistent-storage driver. (For example, if you're on AWS, ensure that your volumes are available in all Availability Zones that your nodes might run in.)
3. Monitor the Worker Group/Fleet pods for volume issues. The faster you can see such issues and react, the more likely that you'll be able to resolve thema.

Uninstall the Infrastructure

To spin down deployed pods, use the helm uninstall command – where `<release-name>` is the namespace you assigned when you installed the chart:

```
helm uninstall <release-name>
```

You can append the `--dry-run` flag to verify which releases will be uninstalled before actually uninstalling them:

```
helm uninstall <release-name> --dry-run
```

Notes on This Example

- If you installed in a namespace, you'll need to include the `-n <namespace>` option in any `helm` command.
- In the above syslog example, you'd still need to configure a corresponding syslog Source in your Cribl Stream Leader, and then commit and deploy it to your Worker Group(s)/Fleet(s).

Known Issues

- The chart currently supports **only** TCP ports in `service` > `ports` for Worker Groups/Fleets. This limitation might be removed in future versions.
- See EKS-specific issues on our GitHub repo.

4.4. Docker Deployment

You can use the following `docker-compose.yml` to stand up a Cribl Stream distributed deployment of a Leader and one or more Workers:

docker-compose.yml

```
version: '3.8'
services:
  master:
    image: cribl/cribl:latest
    environment:
      - CRIBL_DIST_MODE=master
      - CRIBL_DIST_MASTER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@0.0.0.0:4200
      - CRIBL_VOLUME_DIR=/opt/cribl/config-volume
    ports:
      - "19000:9000"
    volumes:
      - "~/cribl-config:/opt/cribl/config-volume"
  workers:
    image: cribl/cribl:latest
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@master:4200
    ports:
      - 9000
```

This uses a local directory, `~/cribl-config`, as a persistent configuration store for Cribl Stream. As a best practice, create this directory on your host OS' filesystem before you run the `docker-compose` command.

If you prefer to use ephemeral storage, simply delete line 8 (the `CRIBL_VOLUME_DIR` definition) and lines 11–12 (the `volumes` configuration) before running the `docker-compose` command.

> Cribl recommends deploying the Leader on stable, highly available infrastructure, because of its role in coordinating all Worker instances.

Deploying Hybrid Workers

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>@main-<Organization-name>.cribl.cloud:4200
```

When you include an auth token in `CRIBL_DIST_MASTER_URL` to deploy managed Cribl Stream or Edge Nodes, it can include only alphanumeric characters or the underscore (`_`). Do not include `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`.

Selecting the Number of Workers

To deploy a Leader Node, plus (e.g.) two Workers already configured and wired up to the Leader, use this command:

```
docker-compose up -d --scale workers=2
```

To deploy a different number of Workers, just change the `workers=2` value.

If you are deploying the Leader and Workers on the same machine or VM, and the Leader is crashing with two workers, make sure you are allocating enough memory to Docker.

Default Image, OS, and Port Assignments

By default, the above command pulls the freshest stable image (tagged `cribl/cribl:latest`) from Cribl's Docker Hub. Our Docker images are currently built on Ubuntu 20.04.6.

Launching Docker containers with the above `docker-compose.yml` file will default to the following URLs and ports:

- Leader URL: `http://localhost:19000`
- Worker URLs: `http://localhost:<automatically-assigned-host-ports>`

With virtual machines, the VMs' IP addresses would replace `localhost`. The automatic assignment of available host-OS ports to the Workers prevents port collisions. **Within** the Docker container, these ports will forward over TCP to port 9000. To see the ports assigned on the OS, enter:

```
docker ps
```

You should see results like these:

```
CONTAINER ID   IMAGE                COMMAND                  CREATED          STATU
a3de9ea8f46f   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 1(
40aa687baefc   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 1(
df362a65f7d1   cribl/cribl:latest   "/sbin/entrypoint.sh…"   13 seconds ago   Up 1:
```

The host-OS ports are shown on the left, forwarding to the container-internal ports on the right. You can use the `docker_workers_N` ports if you want to log directly into Workers. In the above example:

- Worker1 URL: `http://localhost:63411`
- Worker2 URL: `http://localhost:63410`

Adding Port, Forwarding, and Protocol Assignments

To specify additional ports in a distributed deployment, add them to the `docker-compose.yml` file's `workers` > `ports` section:

docker-compose.yml (modified)

```
version: '3.8'
services:
  [...]
  workers:
    image: cribl/cribl:latest
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://${CRIBL_DIST_TOKEN:-criblmaster}@master:4200
    ports:
      - 9000
      - <Additional ports go here>
```

You can specify port forwarding, and TCP versus UDP protocol, in the same keys:

Specifying protocols when adding ports

```
    ports:
      - 9000
      - "10060/tcp"
      - "10070/tcp"
```

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.

Note that the sample `docker-compose.yml` provided above automatically specifies the latest stable image at:

```
image: cribl/cribl:latest
```

You can explicitly pull the latest stable image with this CLI command:

```
docker pull cribl/cribl:latest
```

Updating the Packaged OS

Cribl strongly recommends that you monitor and patch vulnerabilities in the packaged OS. The base OS might have been updated with fixes for new vulnerabilities discovered after Cribl published its container images. This is especially important if you choose to keep an earlier Cribl image in production.

You can update the base OS image by updating the package, as shown in the following `Dockerfile`. This example assumes you're updating Cribl's `latest` image:

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

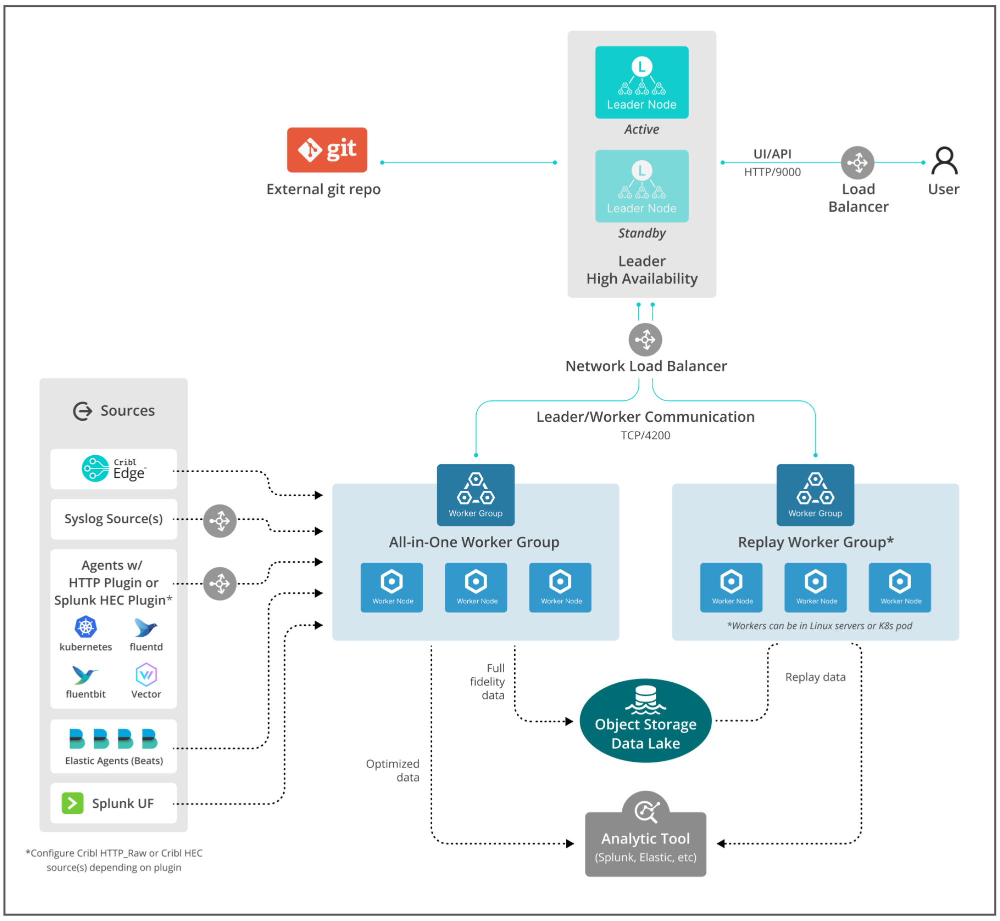
5. Reference Architectures

This section hosts a growing collection of diagrams representing possible Cribl Stream deployment architectures for different use cases. Please see the considerations, details, and disclaimers accompanying each diagram:

- All-in-One Reference Architecture
- Syslog Reference Architecture
- Multiple Agents Reference Architecture
- Comprehensive Reference Architecture
- Comprehensive Hybrid Cloud Reference Architecture

5.1. All-In-One Reference Architecture

This reference architecture shows one possibility for ingesting and replaying diverse data types from multiple senders. The Sources shown at left are just representative examples. The replay workers at right could be hosted on Linux servers, in Kubernetes Pods, etc.



All-in-one reference architecture

Download an editable version of this diagram in SVG or draw.io format. Download Cribl stencils here.

Sizing Considerations

- An all-in one Worker Group works best with fewer than 2000 Sources (agents, syslog senders, etc.).
- Use Cribl's Sizing Calculator to adequately size your own deployment: number of nodes, CPUs, etc.
- Syslog over TCP should be under 200 GB/day from any one sender.
- Account for up to 50% bursts in traffic.

Load Balancers

The load balancer in front of the Leader UI can be an Application Load Balancer or a Network Load Balancer.

The load balancer between the Cribl Workers and the Cribl Leader listening on port 4200 must be a Network Load Balancer or equivalent.

Leader High Availability

Configuring Leader high availability requires a standby server, a load balancer (details here) between users and the Cribl UI port, and a Network Load Balancer between the Cribl Leader and Workers.

Leader high availability is optional. Many customers opt instead for a single Leader, and a remote Git repo for scheduled backups. If the Leader becomes unavailable, you can use the Git repo to provision a new Leader, restoring configuration from a backup.

However, a single Leader is reliable only when Workers are running Cribl Sources that fully leverage a shared-nothing architecture (such as Raw HTTP, Splunk HEC, Elasticsearch API, Amazon S3 Source, etc). Here, Workers can process data even if the Leader is unavailable. A Leader without failover will not reliably support Collectors, which require a Leader's ongoing orchestration.

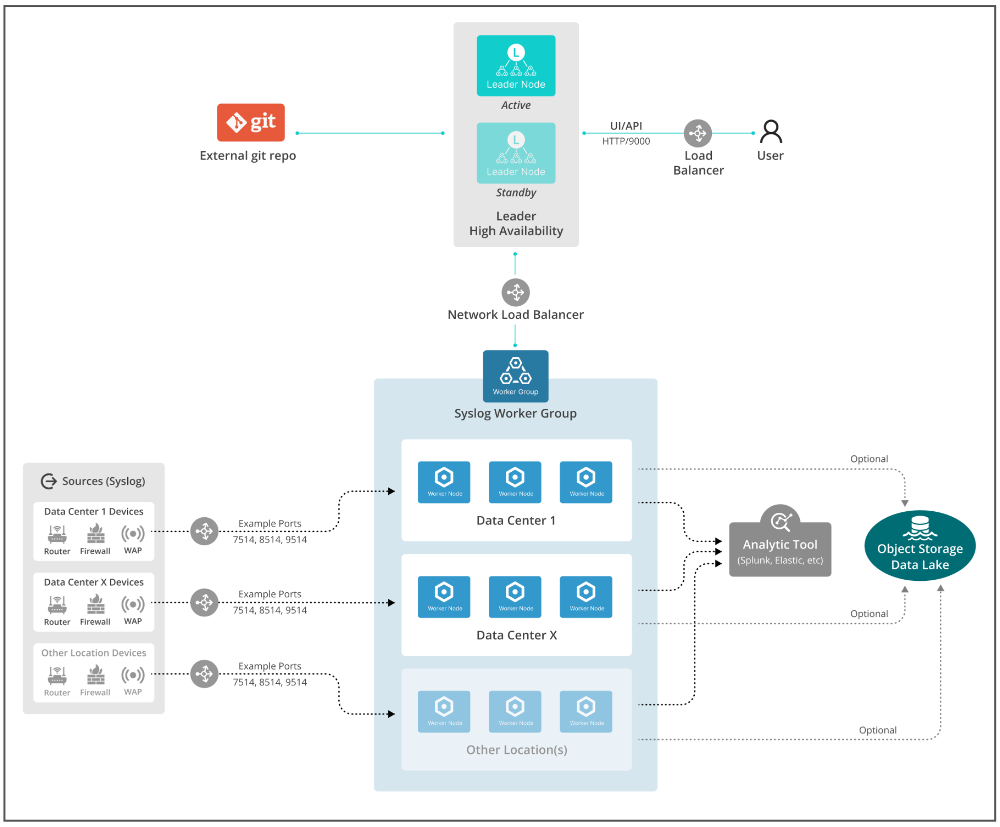
Disclaimers

All product names, logos, brands, trademarks, registered trademarks, and other marks listed in this document are the property of their respective owners. All such marks are provided for identification and informational purposes only. The use of these marks does not indicate affiliation, endorsement, or ownership of the marks or their respective owners.

This document is provided "as is" with NO EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS. It is intended to aid users in displaying system architecture, but might not be applicable or appropriate in all circumstances. You should not act on any information provided until you have formed your own opinion through investigation and research. Cribl is not responsible for any use of this document or the information provided herein.

5.2. Syslog Reference Architecture

This reference architecture shows one Cribl Stream deployment possibility for collecting syslog data from multiple geographic regions. This particular example processes all syslog data through a single Worker Group.



Syslog reference architecture

Download an editable version of this diagram in SVG or draw.io format. Download Cribl stencils here.

Note that the load balancer shown above between the Leader and Worker Group should be dedicated only to communication between the Leader and Workers. At the lower left, the diagram shows separate load balancers dedicated to communication between the syslog senders and the Stream Worker Nodes.

General Sizing Considerations

- Use Cribl's Sizing Calculator to adequately size the number of nodes, CPUs, etc.
- Syslog over TCP should be under 200 GB/day from any one sender.
- Account for up to 50% bursts in traffic.
- General sizing guideline is maximum 200 GB/day throughput per vCPU, or 400 GB/day/CPU core.

- Size for n+50% (to account for bursts and additional high availability).
- Sources that use long-lived TCP connections should be converted to UDP, or to protocols that batch data in different TCP sessions (e.g., most HTTP Push Sources). Sources that send over long-lived TCP connections are subject to “TCP pinning.” They can send up to 200 GB/day (if the Stream Worker is built on CPU cores).

Single or Multiple Worker Groups?

The single-Group architecture shown here is one approach. It might or might not be desirable for your use case.

Pros of using a single Worker Group:

- Commit and deploy only once per change.
- Manage all syslog configuration in one Group.

Cons of using a single Worker Group:

- Additional filtering required if your data needs to be sent to diverse Destinations.

Cribl Source Config Considerations

- Syslog over UDP close to senders is ideal.
- If using Syslog over TCP, a single source can send up to 200 GB/day (if workers are built on CPU cores), or up to 100 GB/day (if workers are built on hyperthreaded vCPUs).
- The Worker Group can also process other data (such as from Filebeat source, Splunk forwarders, HTTP listeners, etc.). However, if the agent count reaches the thousands, a dedicated Worker Group per data type is better.
- Preferred practice: Different upstream technologies should send syslog on different syslog ports (e.g., PAN on 5514, Cisco ASA on 6514, Zscaler on 7514).

Cribl Route Config

Map this Route Filter example by technology: PAN, ASA, Windows, etc. This example assumes a Palo Alto Networks host that includes `PAN` or `FW` in its name:

- `__inputId.startsWith('syslog') && (host.indexOf('FW')> -1 || host.indexOf('PAN')> -1)` -> Palo Alto Pack as Pipeline.
- `__inputId=='PAN_Syslog'` -> Palo Alto Pack as Pipeline.

- `__inputId=='Imperva_Syslog'` -> Imperva Pack as Pipeline.

Cribl Output Router Config

This example shows sending to multiple instances of the same Destination type – data in DC1 should send to Splunk indexers in DC1, data in DC2 should send to Splunk indexers in DC2), etc. Stream's Output Router Destination works well for this use case.

Filter and Destination Examples

- Region==`East`, route to Splunk Virginia indexers.
- Region==`Central`, route Splunk Austin indexers.
- Owner==`Team420`, route to Elastic indexers.

Cribl Load Balancers

The load balancer in front of the Leader UI can be an Application Load Balancer or a Network Load Balancer.

The load balancer between the Cribl Workers and the Cribl Leader listening on port 4200 must be a Network Load Balancer or equivalent.

Third-Party Load Balancer Configs

This section gathers LB configuration examples and resources for several syslog senders.

F5 BIG-IP Example

```
ltm virtual udpsyslog_514_vs {
destination 10.10.10.10:514
ip-protocol udp
mask 255.255.255.255
pool udpsyslog_514_pool
profiles {
    udp { }
}
Vlans-disabled
}

ltm pool udpsyslog_514_pool {
members {
    10.10.20.10:514 {
        address 10.10.20.10
        session monitor-enabled
        state up
    }
    10.10.20.20:514 {
        address 10.10.20.20
        session monitor-enabled
        state up
    }
}
monitor tcp
service-down-action reset
}
```

Citrix Netscaler Example

Reference: Load Balancing syslog Servers.

```
add service service1 192.0.2.10 SYSLOGUDP 514
add service service2 192.0.2.11 SYSLOGUDP 514
add service service3 192.0.2.11 SYSLOGUDP 514
add service service4 192.0.2.10 SYSLOGTCP 514
add service service5 192.0.2.11 SYSLOGTCP 514
add service service6 192.0.2.11 SYSLOGTCP 514
add lb vserver lbvserver1 SYSLOGUDP -lbMethod AUDITLOGHASH
add lb vserver lbvserver1 SYSLOGTCP -lbMethod AUDITLOGHASH
bind lb vserver lbvserver1 service1
bind lb vserver lbvserver1 service2
bind lb vserver lbvserver1 service3
bind lb vserver lbvserver1 service4
bind lb vserver lbvserver1 service5
bind lb vserver lbvserver1 service6
add syslogaction sysaction1 -lbVserverName lbvserver1 -logLevel All
add syslogpolicy syspol1 ns_true sysaction1
bind system global syspol1
```

AWS NLB Guide

Reference: UDP Load Balancing for Network Load Balancer.

NGINX Plus Guide

Reference: TCP and UDP Load Balancing.

HAProxy (Including TCP Break-Apart)

Reference: haproxy speaks syslog.

This example resides at `/usr/local/etc/haproxy/haproxy.cfg`:

```
ring worker1
format rfc5424
maxlen 1500
size 32764
timeout connect 5s
timeout server 10s
server worker1 worker1.example.com:1514
ring worker2
format rfc5424
maxlen 1500
size 32764
timeout connect 5s
timeout server 10s
server worker2 worker2.example.com:1514

# Add as many ring buffers as needed here

log-forward syslog-lb
# UDP Listener (IPv4/IPv6)
dgram-bind :::1514

# TCP Listener (IPv4/IPv6)
bind :::1514

# Load-balance outgoing messages on 2 TCP outputs
log ring@worker1 sample 1:2 local0
log ring@worker2 sample 2:2 local0

maxconn 65536
timeout client 10s
```

Leader High Availability

Configuring Leader high availability requires a standby server, a load balancer (details here) between users and the Cribl UI port, and a Network Load Balancer between the Cribl Leader and Workers.

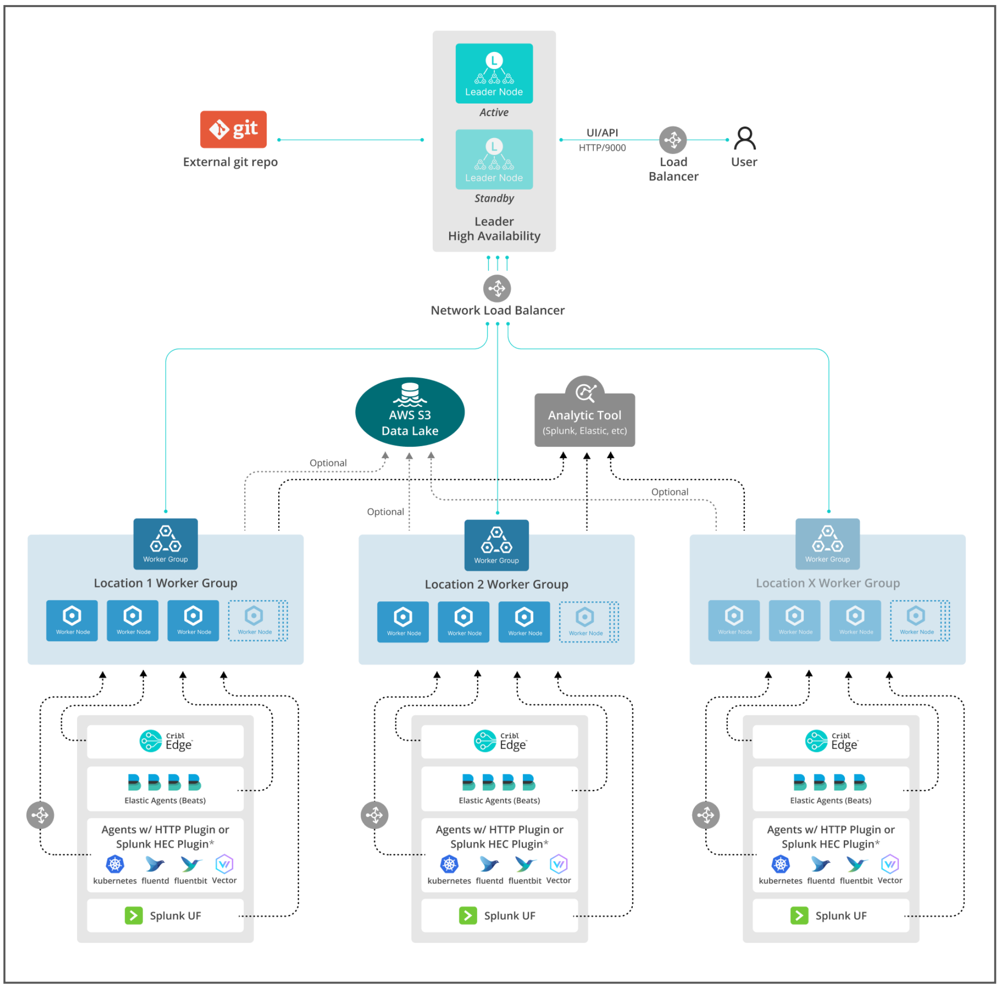
Leader high availability is optional. Many customers opt instead for a single Leader, and a remote Git repo for scheduled backups. If the Leader becomes unavailable, you can use the Git repo to provision a new Leader, restoring configuration from a backup.

However, a single Leader is reliable only when Workers are running Cribl Sources that fully leverage a shared-nothing architecture (such as Raw HTTP, Splunk HEC, Elasticsearch API, Amazon S3 Source, etc). Here, Workers can process data even if the Leader is unavailable. A Leader without failover will not reliably support Collectors, which require a Leader's ongoing orchestration.

Disclaimers

5.3. Multiple Agents Reference Architecture

This reference architecture shows one Cribl Stream deployment possibility for ingesting data from a large quantity of agents. Here, a single basic configuration is replicated across multiple Worker Groups, organized by geographic location.



Multiple agents reference architecture

Download an editable version of this diagram in SVG or draw.io format. Download Cribl stencils here.

Single or Multiple Worker Groups?

The multiple-Groups architecture shown here is one approach. It might or might not be desirable for your use case.

Pros of using multiple Worker Groups:

- Configuration granularity.

- Can use Packs to export/reuse configurations across Groups.

Cons of using multiple Worker Groups:

- Multiple configurations to maintain (even if shared via Packs).

General Sizing Considerations

- Size based on data throughput (in+out), and on number of incoming TCP connections.
- Use Cribl's Sizing Calculator to adequately size the the number of nodes, CPUs, etc., based on throughput.

Sizing Examples

Below are two scenarios demonstrating how to determine the number of Stream Worker Processes you'll need to allocate. Both scenarios assume the same distribution of chatty versus medium-chatty agents, but with a different balance of throughput volume versus agents volume.

How Many Events per Second?

We assume three tiers of "chatty" agents, based on events per second. You'll probably recognize your senders from these definitions:

- Chatty agents (100 EPS/agent) – Size 150 agents/vCPU. (Examples are domain controllers or intermediary agents.)
- Medium-chatty agents (30 EPS/agent) – Size 250 agents/vCPU. (Most servers will fall into this medium category.)
- Low-volume agents (3 EPS/agent) – Size 5000 agents/vCPU (Examples are workstations.)

For the most accurate sizing, obtain EPS reports from your current observability tools.

Using the Scenarios

In each scenario, to size based on the number of agents directly feeding Cribl Stream, take the **largest** vCPU count from the pair of volume versus ports calculations.

Scenario 1

40 TB/day throughput, 40,000 agents, 10% of them very chatty, 90% medium-chatty.

- Sizing for volume (non-hyperthreaded CPU cores): 205 Worker Processes.

- Sizing from TCP ports (non-hyperthreaded CPU cores): (0.10 x 40,000 / 150) + (0.90 x 40,000 / 250) = (27 + 144) = 171.
- Result: Allocate **205** Worker Processes, the larger of the two.

Scenario 2

4 TB/day throughput, 20,000 agents, 10% of them very chatty, 90% medium-chatty.

- Sizing from vCPUs (non hyperthreaded CPU Cores): 11 Worker Processes.
- Sizing from TCP ports (non-hyperthreaded CPU Cores): (0.10 x 20,000 / 150) + (0.90 * 20,000 / 250) = (14 + 72) = 86.
- Result: Allocate **86** Worker Processes, the larger of the two.

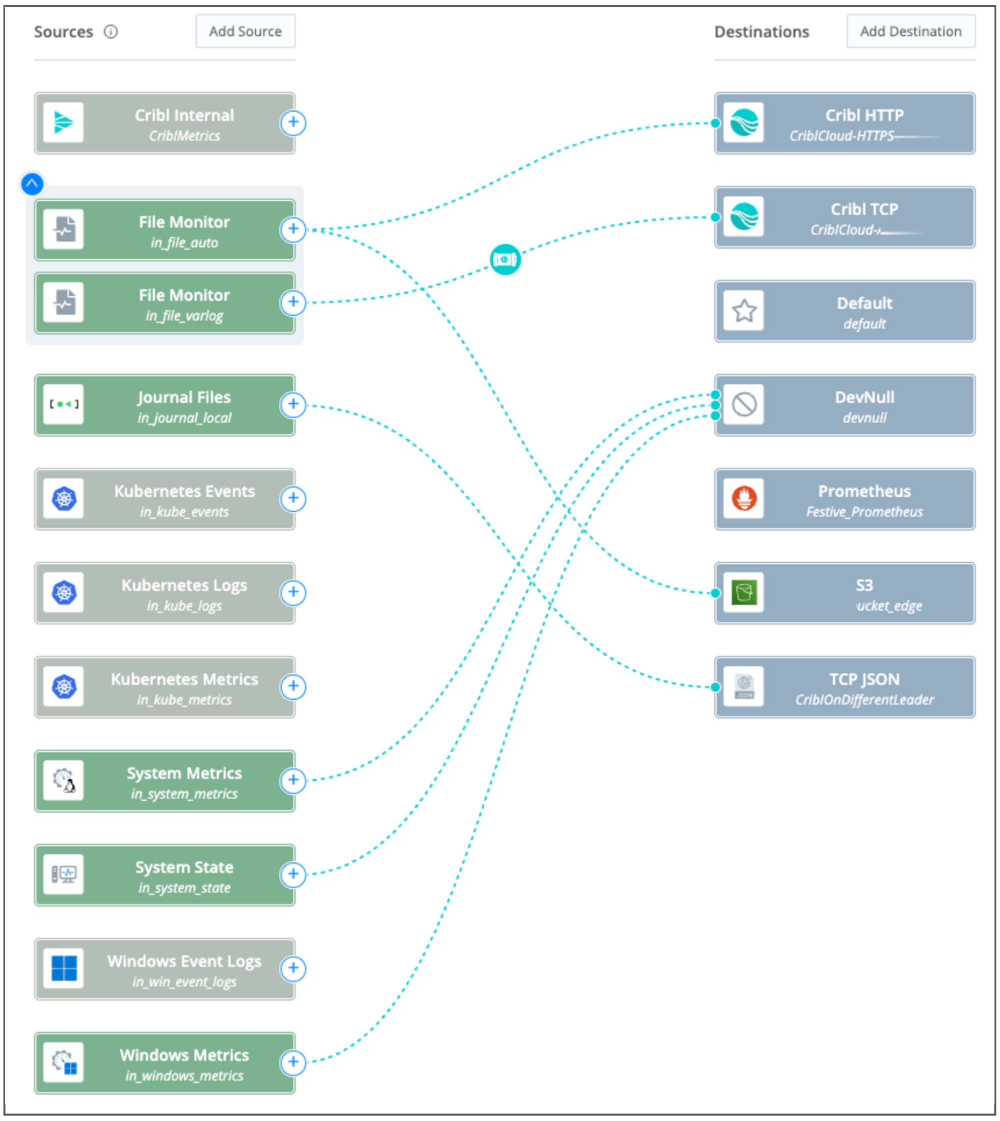
Cribl Edge Configuration

You can configure Cribl Edge to send data to Cribl Stream via a Cribl TCP, Cribl HTTP, or TCP JSON Destination.

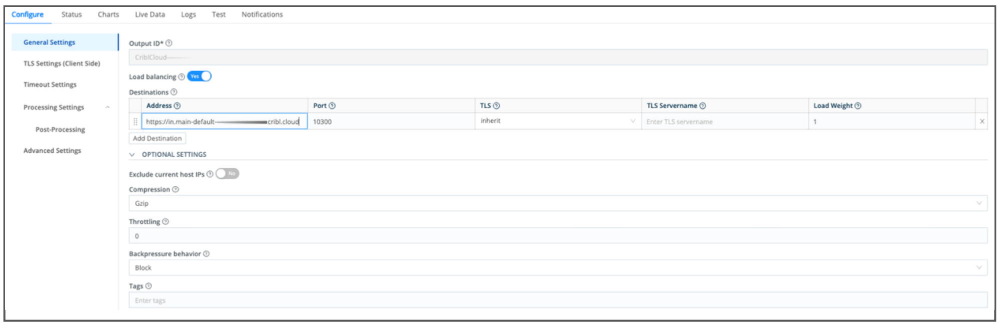
Cribl TCP is ideal for environments where the Edge Nodes and the Cribl Workers are managed by the same Leader. You should always enable this Destination’s load-balancing option. Cribl TCP is the recommended and most performant option. With this option, license metering occurs **only** when the data is collected by the Edge Node, so you are billed only once.

Cribl HTTP is also suitable for environments where the Edge Nodes and the Cribl Workers are managed by the same Leader. Cribl HTTP is ideal for scenarios where the data must traverse a proxy. If you use Cribl HTTP in a non-proxy environment, there will be about a 10% impact on throughput performance, compared to Cribl TCP. The impact will be even larger with an actual proxy in the middle. (Therefore, monitor the throughput impact when traversing a proxy.) You should always enable this Destination’s load-balancing option. With this option, license metering occurs **only** when the data is collected by the Edge Node, so you are billed only once.

TCP JSON is ideal for environments where the Edge Nodes are managed by a **different** Leader than the Stream Workers. Neither Cribl TCP nor Cribl HTTP will work in this scenario. You should always enable this Destination’s load-balancing option.



Edge UI, showing connections to all three Destinations



Configure | Status | Charts | Live Data | Logs | Test | Notifications

General Settings
TLS Settings (Client Side)
Timeout Settings
Processing Settings
Post-Processing
Advanced Settings

Output ID*: CriblCloud-[redacted]

Load balancing: Yes

Destinations

	Address	Port	TLS	TLS Servername	Load Weight
	https://in.main-default-[redacted]cribl.cloud	10300	inherit	Enter TLS servername	1

Add Destination

OPTIONAL SETTINGS

Exclude current host IPs: No

Compression: Gzip

Throttling: 0

Backpressure behavior: Block

Tags: Enter tags

Example Cribl TCP Destination config

Elastic Agent Configuration

The configuration to use in `filebeat.yml` depends on your agent's version.

Versions 8.x+

```
setup.ilm.enabled: false
```

Versions 7.15 Through 7.17

```
output.elasticsearch:
hosts: ["https://<your-host-here>.com:9200/elastic"]
```

Or:

```
output.elasticsearch:
hosts: ["<server-name-here>:9200/elastic", "<another-server-name-here>:9200/elastic
```

Versions 7.14 and Earlier

```
output.elasticsearch:
hosts: ["[https://<your-host-here>.com:9200](https://<your-host-here>.com:9200)"]
```

Or:

```
output.elasticsearch:
hosts: ["<server-name-here>:9200", "<another-server-name-here>:9200"]
```

Configure the agent's `worker` setting to avoid TCP pinning. See Configure the Elasticsearch Output.

Splunk Forwarder Configuration

Here are configurations for a few different scenarios.

Setting `blockOnCloning` to `false` ensures that if either Cribl or Splunk becomes unavailable, the other system will continue receiving data without dropping events, preventing potential data loss or interruptions in transmission.

Send All Data to Cribl

`outputs.conf` to send to both Splunk and Cribl:

```
[tcpout]
defaultGroup=splunk,stream
blockOnCloning=false

[tcpout:splunk]
server=10.1.1.197:9997,10.1.1.198:9997
blockOnCloning=false

[tcpout:stream]
server=10.1.1.200:9997,10.1.1.201:9997
sendCookedData=true
blockOnCloning=false
```

`outputs.conf` to send only to Cribl:

```
[tcpout]
defaultGroup=stream
blockOnCloning=false

[tcpout:stream]
server=10.1.1.200:9997,10.1.1.201:9997
sendCookedData=true
blockOnCloning=false
```

Send All Data (Minus Splunk Internal Logs) to Cribl

`inputs.conf` for internal logs:

```
# datasource1.log only gets sent to Splunk

[monitor:/path/to/log/datasource1.log]
_TCP_ROUTING = splunk
index = ds1_index
sourcetype = sourcetype1
```

`inputs.conf` for all other logs:

```
# datasource2.log will get sent to both Splunk and Stream

[monitor:/path/to/log/datasource2.log]
_TCP_ROUTING = splunk, stream
index = ds2_index
sourcetype = sourcetype2
```

`outputs.conf` for internal logs:

```
[tcpout]
defaultGroup=splunk,stream
blockOnCloning=false

[tcpout:splunk]
server=10.1.1.197:9997,10.1.1.198:9997
blockOnCloning=false

[tcpout:stream]
server=10.1.1.200:9997,10.1.1.201:9997
sendCookedData=true
blockOnCloning=false
```

Fluent Bit Configuration

Here are sample configurations for three different ways to ingest Fluent Bit data.

Splunk HEC Source on Cribl Workers

Fluent Bit reference

```
[OUTPUT]
Name cribl
Match *
host <HOST>
port 8088
Cribl_HEC_Token xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx
Cribl_HEC_Send_Raw On #optional: to raw instead of event endpoint
TLS On
TLS.Verify Off #optional
```

Raw HTTP Source on Cribl Workers

Fluent Bit reference

```
[OUTPUT]
Name http
Match *
Host cribl_worker_load_balanced_vip
Port 10443
URI /optional/uri/for/filtering/in/cribl
Format json_lines
Header Authorization <Cribl-generated token>
Header X-Key-B Value_B
```

TCP JSON Source on Cribl Workers

Fluent Bit reference

```
[OUTPUT]
Name tcp
Format json_lines
Workers 2
tls on #optional for tls setting
json_date_key date
json_date_format epoch

# Note: Use mTLS, because this plugin doesn't appear to support authTokens
```

Cribl Destination Considerations

Adjust Cribl Destinations' **Advanced Settings** > **Max connections** setting (where available) to limit the load on receivers. E.g.: For a Splunk Load Balanced Destination, if you have 5 Worker Groups sending data to the same Splunk indexer cluster, with 25 indexers, set **Max connections** to between `5-10`.

Load Balancers

The load balancer in front of the Leader UI can be an Application Load Balancer or a Network Load Balancer.

The load balancer between the Cribl Workers and the Cribl Leader listening on port 4200 must be a Network Load Balancer or equivalent.

Leader High Availability

Configuring Leader high availability requires a standby server, a load balancer (details here) between users and the Cribl UI port, and a Network Load Balancer between the Cribl Leader and Workers.

Leader high availability is optional. Many customers opt instead for a single Leader, and a remote Git repo for scheduled backups. If the Leader becomes unavailable, you can use the Git repo to provision a new Leader, restoring configuration from a backup.

However, a single Leader is reliable only when Workers are running Cribl Sources that fully leverage a shared-nothing architecture (such as Raw HTTP, Splunk HEC, Elasticsearch API, Amazon S3 Source, etc). Here, Workers can process data even if the Leader is unavailable. A Leader without failover will not reliably support Collectors, which require a Leader's ongoing orchestration.

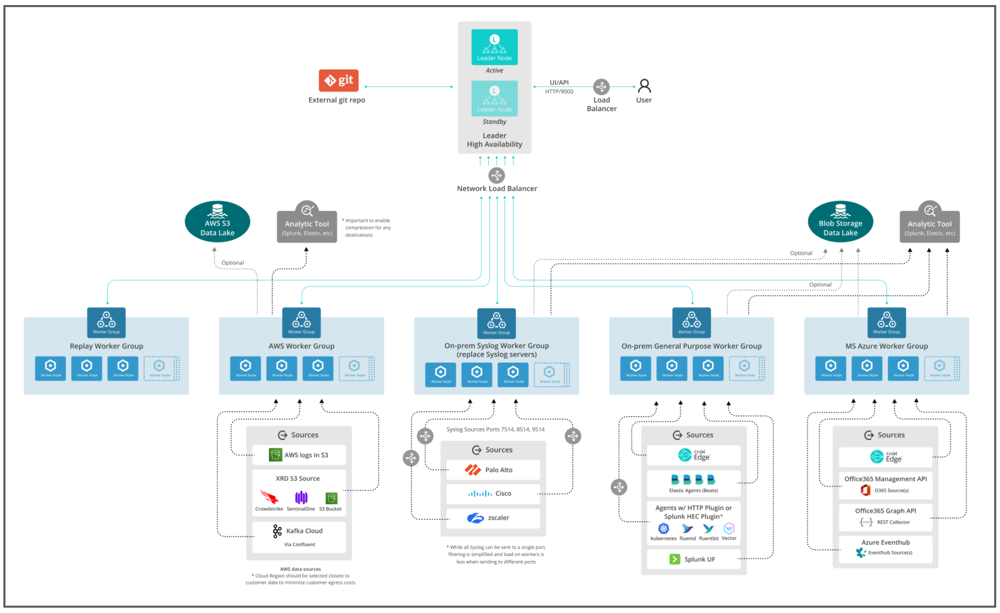
Disclaimers

All product names, logos, brands, trademarks, registered trademarks, and other marks listed in this document are the property of their respective owners. All such marks are provided for identification and informational purposes only. The use of these marks does not indicate affiliation, endorsement, or ownership of the marks or their respective owners.

This document is provided "as is" with NO EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS. It is intended to aid users in displaying system architecture, but might not be applicable or appropriate in all circumstances. You should not act on any information provided until you have formed your own opinion through investigation and research. Cribl is not responsible for any use of this document or the information provided herein.

5.4. Comprehensive Reference Architecture

This reference architecture shows one Cribl Stream deployment possibility for sending data to a diverse mix of destinations, such as data centers, while minimizing data egress costs. The multiple Worker Groups might be organized functionally, or by geographic location.



Comprehensive reference architecture

Download an editable version of this diagram in SVG or draw.io format. Download Cribl stencils here.

This architecture shows customer-managed (on-prem) Worker Groups and primary/standby Leaders. For a Cribl.Cloud-centric alternative, see Comprehensive Hybrid Cloud Reference Architecture.

Division of Labor

In this architecture, you manage all components.

You also manage a remote Git repo. You manage commit, push, deploy, and backup operations through the Cribl Leader.

The Replay Worker Group can be on-demand, and provisioned when heavy replays are planned.

General Sizing Considerations

- Size based on data throughput (in+out), and on number of incoming TCP connections.

- Use Cribl's Sizing Calculator to adequately size the number of nodes, CPUs, etc., based on throughput.

Destination Considerations

Adjust Destinations' **Advanced Settings** > **Max connections** setting (where available) to limit the load on receivers. E.g.: For a Splunk Load Balanced Destination, if you have 5 Worker Groups sending data to the same Splunk indexer cluster, with 25 indexers, set **Max connections** to between `5-10`.

Load Balancers

The load balancer in front of the Leader UI can be an Application Load Balancer or a Network Load Balancer.

The load balancer between the Cribl Workers and the Cribl Leader listening on port 4200 must be a Network Load Balancer or equivalent.

Leader High Availability

Configuring Leader high availability requires a standby server, a load balancer (details here) between users and the Cribl UI port, and a Network Load Balancer between the Cribl Leader and Workers.

Leader high availability is optional. Many customers opt instead for a single Leader, and a remote Git repo for scheduled backups. If the Leader becomes unavailable, you can use the Git repo to provision a new Leader, restoring configuration from a backup.

However, a single Leader is reliable only when Workers are running Cribl Sources that fully leverage a shared-nothing architecture (such as Raw HTTP, Splunk HEC, Elasticsearch API, Amazon S3 Source, etc). Here, Workers can process data even if the Leader is unavailable. A Leader without failover will not reliably support Collectors, which require a Leader's ongoing orchestration.

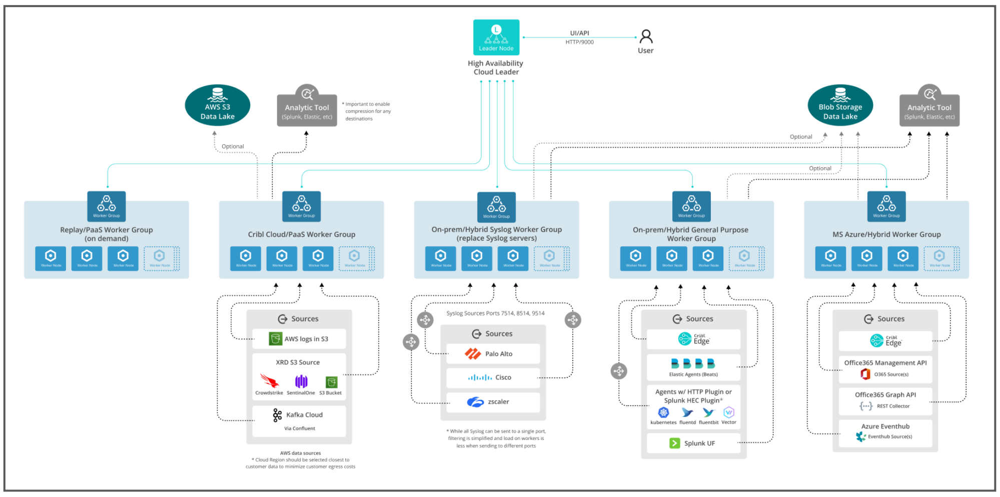
Disclaimers

All product names, logos, brands, trademarks, registered trademarks, and other marks listed in this document are the property of their respective owners. All such marks are provided for identification and informational purposes only. The use of these marks does not indicate affiliation, endorsement, or ownership of the marks or their respective owners.

This document is provided "as is" with NO EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS. It is intended to aid users in displaying system architecture, but might not be applicable or appropriate in all circumstances. You should not act on any information provided until you have formed your own opinion

5.5. Comprehensive Hybrid Cloud Reference Architecture

This reference architecture shows one Cribl Stream deployment possibility for sending data to a mix of analytics tools and data lakes. Data is processed by a mix of Cribl-managed/PaaS (platform as a service) and hybrid Worker Groups, all coordinated by a Cribl.Cloud Leader with failover. The multiple Worker Groups are organized functionally, by Destination type. One Cribl-managed Group is reserved for on-demand replay from data lakes.


Leader Node
High Availability Cloud Leader
UI/API
HTTP/9000
User
AWS S3 Data Lake
Analytic Tool (Splunk, Elastic, etc)
* Important to enable compression for any destinations
Blob Storage Data Lake
Analytic Tool (Splunk, Elastic, etc)
Optional
Optional
Optional
Worker Group
Replay/PaaS Worker Group (on demand)
Cribl Cloud/PaaS Worker Group
On-prem/Hybrid Syslog Worker Group (replace Syslog servers)
On-prem/Hybrid General Purpose Worker Group
MS Azure/Hybrid Worker Group
Worker Node
Sources
AWS logs in S3
XRD S3 Source
Crowdstrike
SentinelOne
S3 Bucket
Kafka Cloud
Via Confluent
AWS data sources
* Cloud Region should be selected closest to customer data to minimize customer egress costs
Syslog Sources Ports 7514, 8514, 9514
Sources
Palo Alto
Cisco
zscaler
* While all Syslog can be sent to a single port, filtering is simplified and load on workers is less when sending to different ports
Sources
Cribl Edge
Elastic Agents (Beats)
Agents w/ HTTP Plugin or Splunk HEC Plugin*
kubernetes
fluentd
fluentbit
Vector
Splunk UF
Sources
Cribl Edge
Office365 Management API
O365 Source(s)
Office365 Graph API
REST Collector
Azure Eventhub
Eventhub Source(s)

Comprehensive hybrid Cribl.Cloud reference architecture

Download an editable version of this diagram in SVG or draw.io format. Download Cribl stencils here.

This architecture shows Cribl.Cloud Leaders with a mix of Cribl-managed and customer-managed (hybrid) Workers. For a fully customer-managed (on-prem) alternative, see Comprehensive Reference Architecture.

Division of Labor

Cribl.Cloud maintains high-availability Leader infrastructure on your behalf.

Cribl.Cloud also maintains secure copies of the Leader's configurations for all Worker Groups (Cribl-managed/PaaS and hybrid).

Even on Cribl-managed/PaaS Worker Groups, configuring and administering Pipelines, Routes, etc., remains your responsibility.

General Sizing Considerations

- Size based on data throughput (in+out), and on number of incoming TCP connections.
- Use Cribl's Sizing Calculator to adequately size the number of nodes, CPUs, etc., based on throughput.

Destination Considerations

On Cribl-managed/PaaS Groups, enable compression on all Destinations that support it, to minimize data egress volume and costs.

Adjust Destinations' **Advanced Settings** > **Max connections** setting (where available) to limit the load on receivers. E.g.: For a Splunk Load Balanced Destination, if you have 5 Worker Groups sending data to the same Splunk indexer cluster, with 25 indexers, set **Max connections** to between `5-10`.

Disclaimers

All product names, logos, brands, trademarks, registered trademarks, and other marks listed in this document are the property of their respective owners. All such marks are provided for identification and informational purposes only. The use of these marks does not indicate affiliation, endorsement, or ownership of the marks or their respective owners.

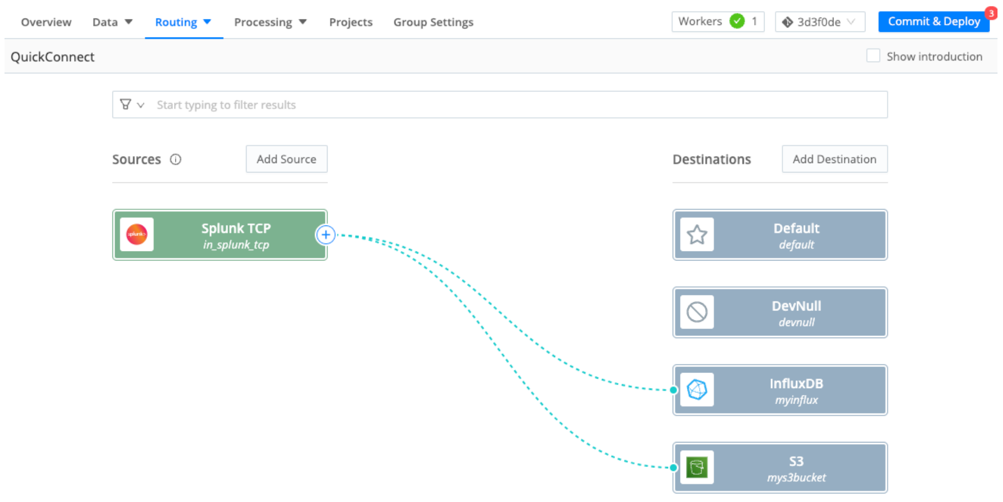
This document is provided "as is" with NO EXPRESS OR IMPLIED WARRANTIES OR REPRESENTATIONS. It is intended to aid users in displaying system architecture, but might not be applicable or appropriate in all circumstances. You should not act on any information provided until you have formed your own opinion through investigation and research. Cribl is not responsible for any use of this document or the information provided herein.

6. QuickConnect

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

You can insert Pipelines or Packs into the connections, to take advantage of Cribl Stream's full range of data-transformation 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 Stream. 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.)

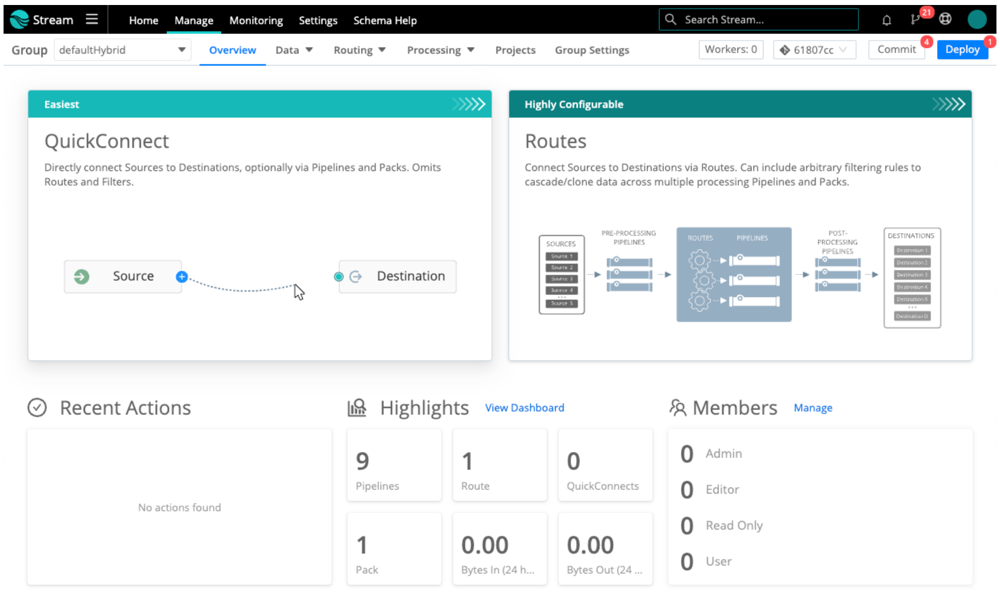


QuickConnect UI

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

When you display the home page of a Cribl Stream single instance or Worker Group, you'll see tiles that prompt you to choose between the **QuickConnect** versus **Route** configuration UIs. Click the left tile to start using QuickConnect.



QuickConnect

Directly connect Sources to Destinations, optionally via Pipelines and Packs. Omits Routes and Filters.

Routes

Connect Sources to Destinations via Routes. Can include arbitrary filtering rules to cascade/clone data across multiple processing Pipelines and Packs.

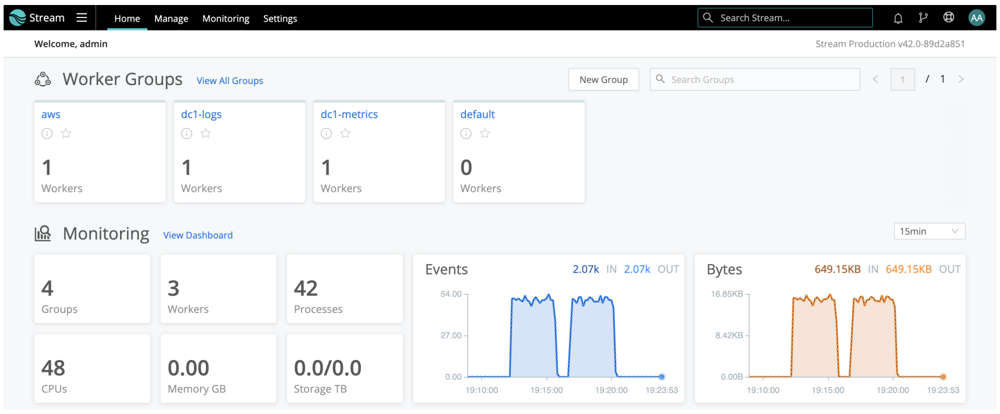
No actions found

9 Pipelines | 1 Route | 0 QuickConnects
1 Pack | 0.00 Bytes In (24 h... | 0.00 Bytes Out (24 ...

0 Admin
0 Editor
0 Read Only
0 User

QuickConnect versus Routing UIs

To display the above landing page in a distributed deployment, you must first select a Group. Click the top nav's **Manage** link to select the `default` or a different Group; or click a Group's tile on the distributed landing page, as shown below.



aws: 1 Workers
dc1-logs: 1 Workers
dc1-metrics: 1 Workers
default: 0 Workers

4 Groups | 3 Workers | 42 Processes
48 CPUs | 0.00 Memory GB | 0.0/0.0 Storage TB

Selecting a Group

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

QuickConnect Versus Routes

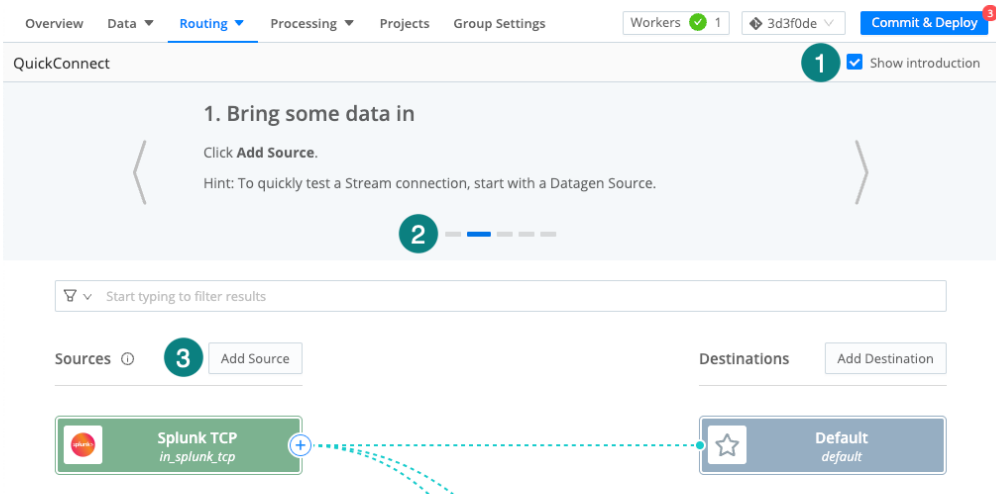
QuickConnect provides access to most of Cribl Stream's configuration options, with a few restrictions (in exchange for the simplified visual interface):

- 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 Stream Routing table's options to clone and cascade data across Pipelines.

- Most Sources are available to fully configure. However, you'll need to use **Routing** > **Data Routes** to configure Collector Sources.
- 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, with **Introduction** help text available at the top of the UI. Here's a little extra help on help, keyed to the orange, numbered callouts in the screenshot below:



QuickConnect Introduction/help carousel

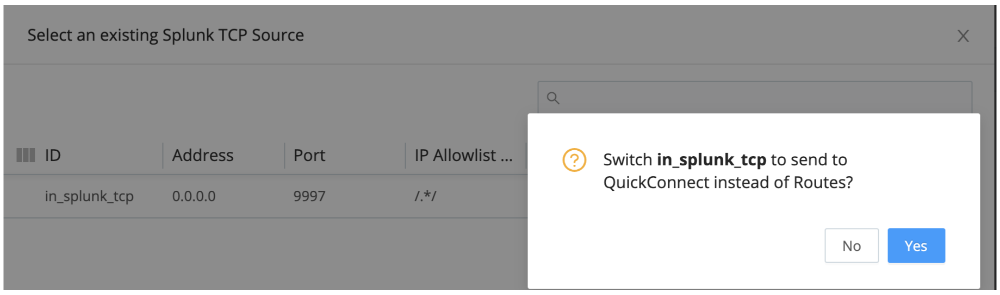
1. Use the **Show introduction** check box at the upper right to toggle the help text on and off.
2. Use the dashed buttons to advance or rewind the help text's carousel (whose position is independent of your connection state).
3. Once you've added at least one Source, the **New Source** button (mentioned in the help text) becomes an **Add Source** button beside the **Sources** header.

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 Stream – 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.



Switch **in_splunk_tcp** to send to QuickConnect instead of Routes?

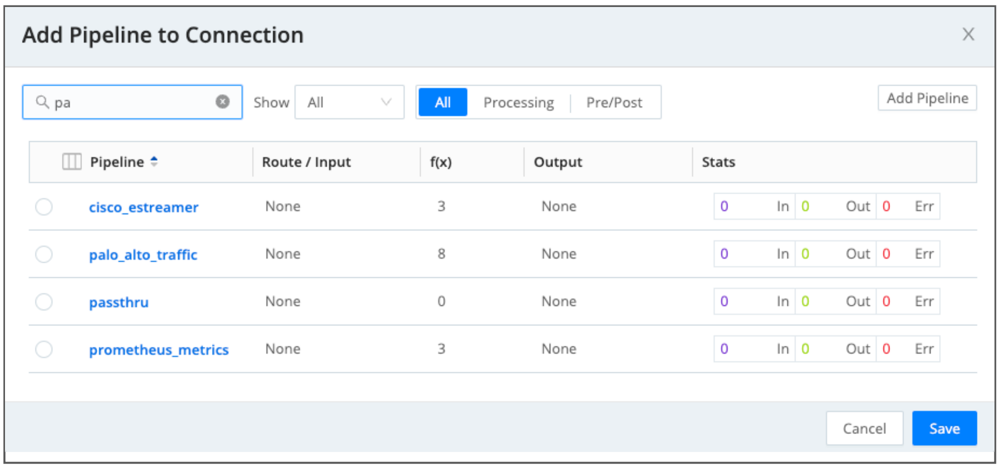
Switching a Source/Destination between worlds

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.

Selecting a Pipeline or Pack

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



Pipeline	Route / Input	f(x)	Output	Stats
cisco_estreamer	None	3	None	0 In 0 Out 0 Err
palo_alto_traffic	None	8	None	0 In 0 Out 0 Err
passthru	None	0	None	0 In 0 Out 0 Err
prometheus_metrics	None	3	None	0 In 0 Out 0 Err

Adding a Pipeline

Here, you're using the radio buttons at left to select 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 new modal, which might offer more configuration than you bargained for.)

Instead, click the 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 modal.

If you've attached Tags to your Pipelines or Packs, you can use the filter field to search against them. Use the format: **TAGS/tag_name**.

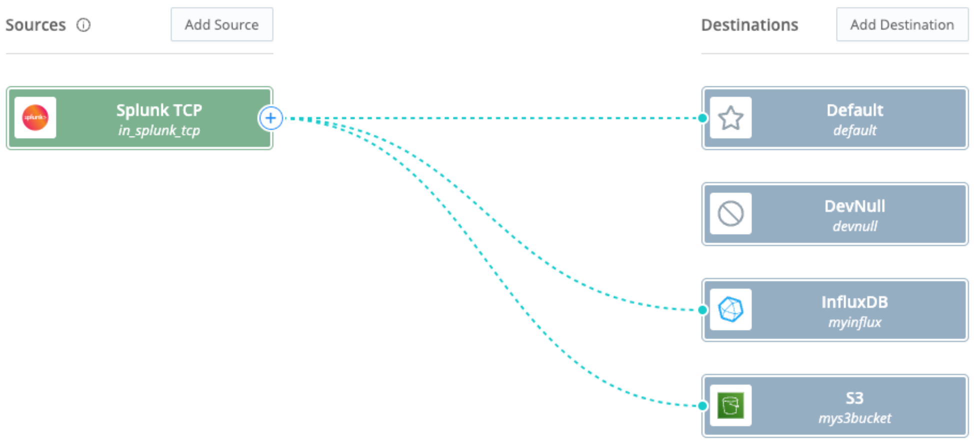
Editing a Pipeline

If you want to modify a Pipeline, this is where you do want to click its blue link in the **Add Pipeline to Connection** modal. For the options available here, see Pipelines.

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.)

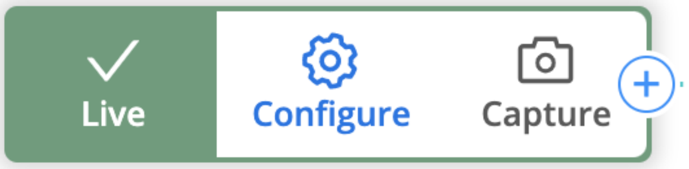
Also, you can connect one Source to multiple Destinations, or multiple Sources to one Destination. Let's isolate examples of both, from the screen capture we started with:



Multiple connections out, multiple connections in

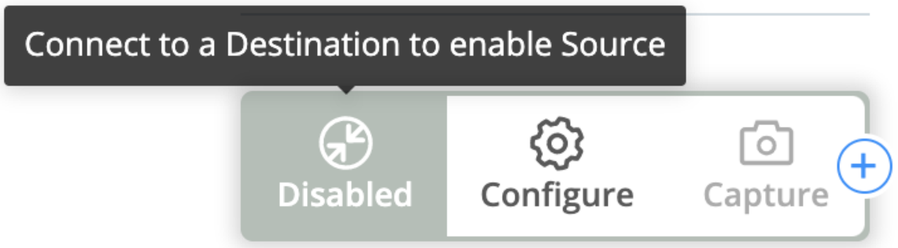
Source/Destination Tiles' Hover Options

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.


Connect to a Destination to enable Source

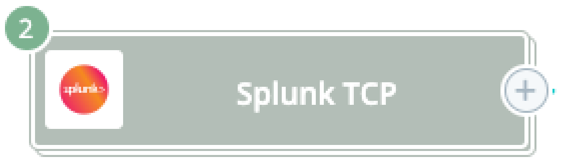
This Source is not misconfigured – it's just lonely

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. (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.



2 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.

7. Workspaces

Workspaces offer a multi-tenancy capability, enabling you to create multiple isolated instances in your Cribl.Cloud Organization. This lets you strengthen your security, and fulfill compliance and isolation requirements.

Each Workspace offers a dedicated virtual public cloud (VPC) that acts as a separate environment within your Organization.

You can manage all your Workspaces in one interface that contains access controls for granting Members and Teams access to individual Workspaces.

Workspaces require a Cribl.Cloud Enterprise plan.

An example use case scenario for creating multiple Workspaces is setting up separate environments for different business units in an enterprise. Data management, security, development, and any other units can have their own federated Workspaces, with completely separate members and permissions lists. This way, they can work in isolation without risk of interference and with increased focus.

Opt-In to Workspaces

Access to configuring multiple Workspaces is available once you opt in to a new Cribl.Cloud UI:

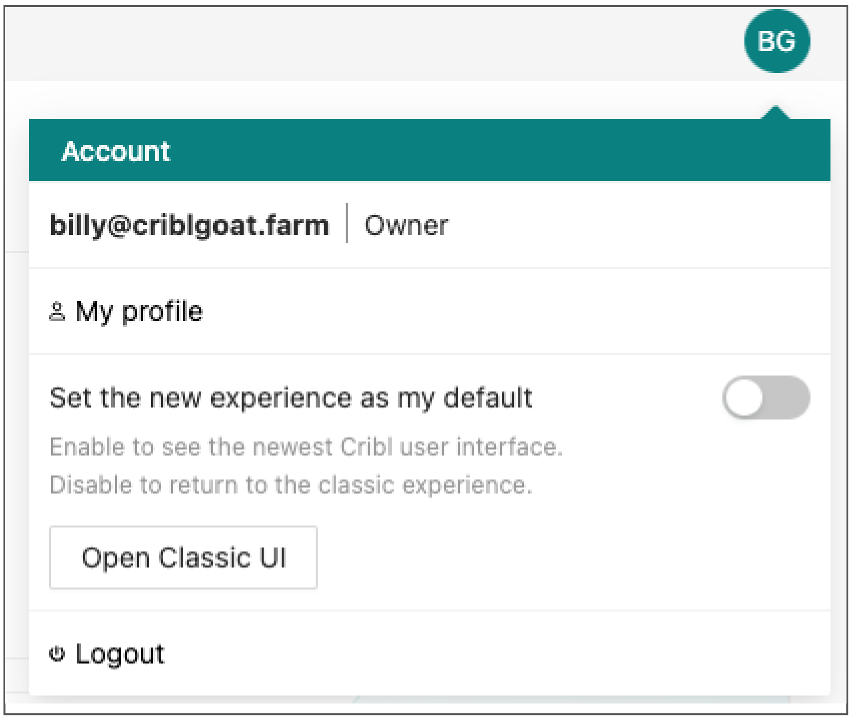
1. On the Cribl.Cloud top bar, click **Try the New Cribl.Cloud**.
2. In the modal, confirm with **Get Started**. You will move to a new user interface for managing Cribl.Cloud, including Workspaces.

Your Cribl.Cloud Organization will open with the new UI every time you re-enter it. You can change this behavior in the following way:

1. Open your user menu at the right side of the top menu.
2. Disable **Set the new experience as my default**.

If you want to return to the old UI:

1. Open your user menu at the right side of the top menu.
2. Disable **Set the new experience as my default**.
3. Click **Open Classic UI**.


BG
Account
billy@criblgoat.farm | Owner
My profile
Set the new experience as my default
Enable to see the newest Cribl user interface.
Disable to return to the classic experience.
Open Classic UI
Logout

You can return to the classic UI at any time

Only users with the **Organization Owner** or **Organization Admin** Member permission can opt in to the new user interface.

Limitations

You can create up to 5 Workspaces per Organization.

Workspaces require a Cribl.Cloud Enterprise plan.

7.1. Configuring Workspaces

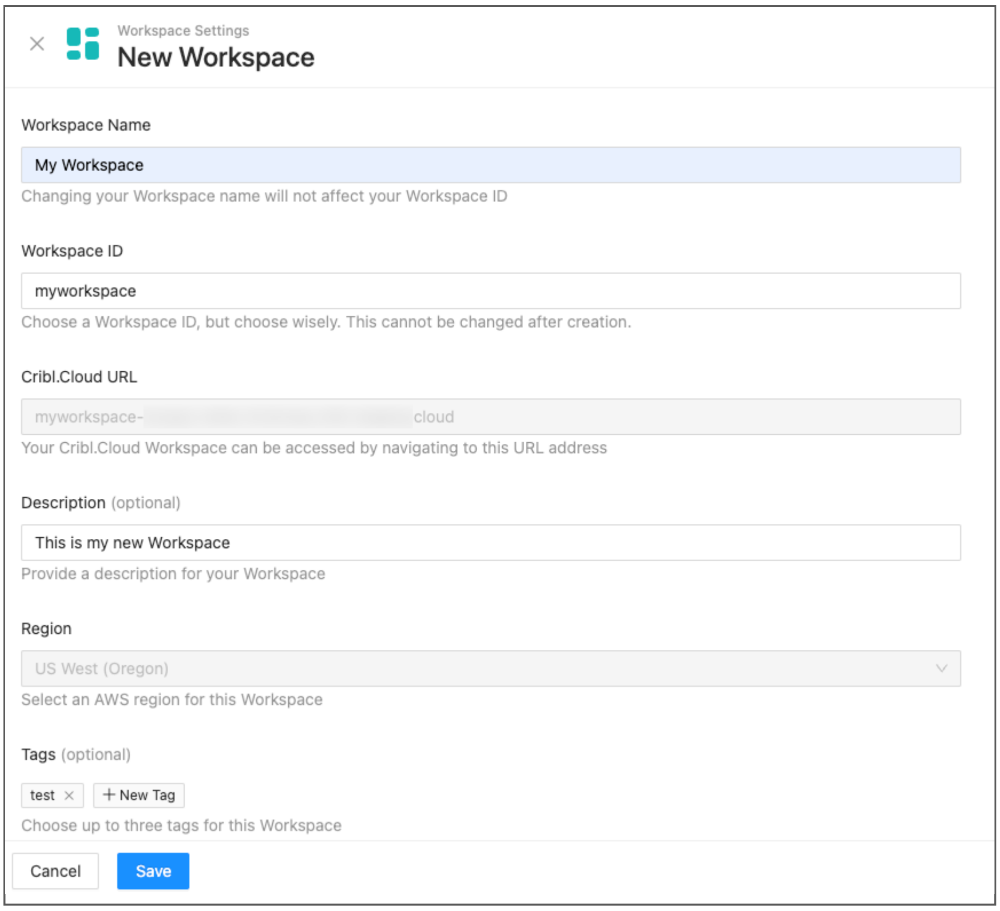
Add New Workspace

Before you try creating a new Workspace, make sure you have opted in to the new Cribl.Cloud UI.

To create a new Workspace:

1. From the top bar of your Organization's front page, click your current Workspace name. This opens a modal with a list of all Workspaces you have access to.
2. Select **Add Workspace**.
3. Fill the form with the following information:

Field	Description	Notes
Workspace Name	Human-readable name for the Workspace.	
Workspace ID	Internal ID for the Workspace. This ID will be part of the Workspace's Cribl.Cloud URL.	**Cannot be changed** once the Workspace is created.
Description	Additional Workspace description.	Optional
Region	Set automatically to correspond to the Organization region.	
Tags	Up to 3 tags you can use to filter Workspaces.	Optional


Workspace Settings

New Workspace

Workspace Name

My Workspace

Changing your Workspace name will not affect your Workspace ID

Workspace ID

myworkspace

Choose a Workspace ID, but choose wisely. This cannot be changed after creation.

Cribl.Cloud URL

myworkspace-[redacted]cloud

Your Cribl.Cloud Workspace can be accessed by navigating to this URL address

Description (optional)

This is my new Workspace

Provide a description for your Workspace

Region

US West (Oregon)

Select an AWS region for this Workspace

Tags (optional)

test × + New Tag

Choose up to three tags for this Workspace

Creating a new Workspace

1. Confirm with **Save**.

The Workspace can take up to several minutes to spin up. While the process continues, you can navigate away and keep working with your Cribl.Cloud instance as usual.

Edit a Workspace

You can edit your Workspace at any time, by clicking the gear ( ) button in the top right corner. You can change the name, description, and tags of a Workspace. However, you can't change its identifier once it has been created.

Delete a Workspace

To delete a Workspace:

1. Click the gear ( ) button in the top right corner of the Workspace's screen.
2. Select **Delete Workspace**.

Only Members with the Organization Owner permission can delete Workspaces.

7.2. Members And Teams

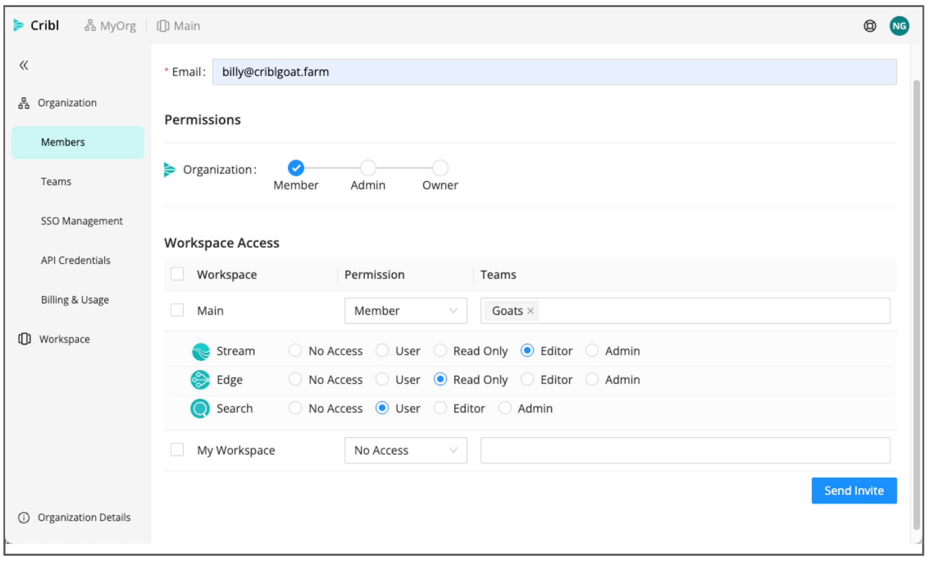
One of the main benefits of Workspaces is the ability to granularly control which users have access to which parts of the system. Access management is done though assigning permissions to individual Members and Teams within a Workspace.

If a Member has overlapping assigned permissions, the highest permission is given.

Members

When inviting a new user to your Organization, configure permissions for Workspaces in the **Workspace Access** section:

1. Select a Member permission in the **Permission** dropdown next to the Workspace.
   When you choose the **Member** permission, configure access for each Cribl product separately:
2. Select **Teams** to assign to the Member.



* Email: billy@criblgoat.farm

Permissions

Organization: [✓ Member] Admin Owner

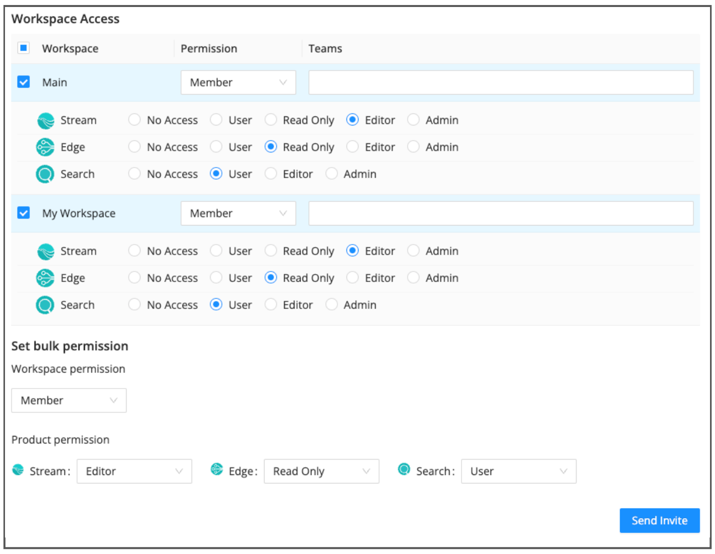
Workspace Access

Workspace	Permission	Teams
Main	Member	Goats ×
My Workspace	No Access	

Stream	No Access	User	Read Only	[✓ Editor]	Admin
Edge	No Access	User	[✓ Read Only]	Editor	Admin
Search	No Access	[✓ User]	Editor	Admin	

Selecting product-level permissions and a Team for a new Member.

Alternatively, select the check box next to one or more Workspaces and choose a permission under **Set bulk permission**. This permission will then be granted to the user for all selected Workspaces.


Workspace Access

Workspace	Permission	Teams
✓ Main	Member	
Stream	No Access · User · Read Only · [✓ Editor] · Admin	
Edge	No Access · User · [✓ Read Only] · Editor · Admin	
Search	No Access · [✓ User] · Editor · Admin	
✓ My Workspace	Member	
Stream	No Access · User · Read Only · [✓ Editor] · Admin	
Edge	No Access · User · [✓ Read Only] · Editor · Admin	
Search	No Access · [✓ User] · Editor · Admin	

Set bulk permission

Workspace permission

Member

Product permission

Stream: Editor   Edge: Read Only   Search: User

Setting bulk permissions for two Workspaces.

Teams

Teams are groups of Members who share the same Workspace and product-level permissions. This allows you to efficiently manage access control by assigning permissions to the Team rather than configuring each Member individually.

To configure a Team:

1. On the left-side menu, expand **Organization** and then select **Teams**.
2. Select **Create Team** and the configuration modal is displayed.
3. Provide a meaningful **Name** and optionally a **Description**.
4. Select the desired Workspace and access to assign to the Team. **Members** allow granular product-level permissions.
5. In **Team Members**, select the Members you want to add, then **Add** them. A Member must already exist to be added to a Team.
6. The table is updated to show the Members you've just added.
7. When finished, **Save** your new Team.

The main Teams page displays the Teams configured on your Workspace. You can **Edit** or delete them as needed from the **Actions** column.

8. Projects

Projects (also called Stream Projects) diversify who can use Cribl Stream, and how:

- Projects create isolated spaces for teams and users to share and manage their data. Project editors get a streamlined visual UI.
- Cribl admins can filter users' access to specific data and Destinations – scoping the access to match what's relevant to those users' work.

This self-service approach to Cribl Stream data can benefit its immediate users, by offering them accelerated access to relevant data, with minimal configuration requirements. It can also benefit their peers.

Project editors get isolated workspaces, and they can customize these workspaces to meet their needs, with no impact on other Cribl Stream users across their organization. Data can be forked into (for example) full-fidelity versus redacted views, to match different groups' authorization and needs.

Breaking Change

Any Projects you created in v.4.1.x are 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.

In Cribl Stream 4.2 and later, Projects/Subscriptions rely entirely on the fine-grained Members/Permissions access control first introduced in Stream 4.2. See Adding Users to Projects for details.

Use-Case Scenario

Imagine that a Cribl administrator wants to filter data from a shared set of firewall Sources, splitting the data between her organization's Security and Ops teams. A requirement is that each team's data should be isolated, with neither team having access to the other team's data. The Security team uses Splunk, while the Operations team uses Elasticsearch.

Projects enable this filtering and isolation. Each team gets secure access to its own sets of data, and each team's data transformations and configuration changes don't affect the other team's data or configs.

Requirements

Enabling Projects requires either an Enterprise license and a distributed on-prem deployment, or a Cribl.Cloud Enterprise plan.

Projects' Subscriptions can ingest data only from Cribl Stream Sources configured as **Send to Routes**, not from QuickConnect Sources. (This is despite Projects' QuickConnect-like Project view, introduced below.)

Components

This feature relies on the following building blocks:

- Subscriptions
- Data Projects
- Roles and Permissions

Data Projects

The **Data Projects** tab is where Cribl admins can add and configure Projects, share access to those Projects, and associate Subscriptions with specific Destinations.

Admins see an expanded version of the same drag-and-drop visual UI that Project editors can use to connect configured Subscription with configured Destinations.

Subscriptions

Each Subscription filters a subset of a Worker Group's incoming data to expose to Project editors, along with processing options for that data.

Members and Permissions

In Cribl Stream 4.2 and later, Projects work with the Members and Permissions authorization system first introduced in Stream 4.2. Cribl Stream admins – users who hold the Stream- or Worker Group-level `Admin` or `Editor` Permission, or the legacy `stream_admin` Role – can create and configure Projects and Subscriptions. They can access resources outside individual Projects to build the Projects.

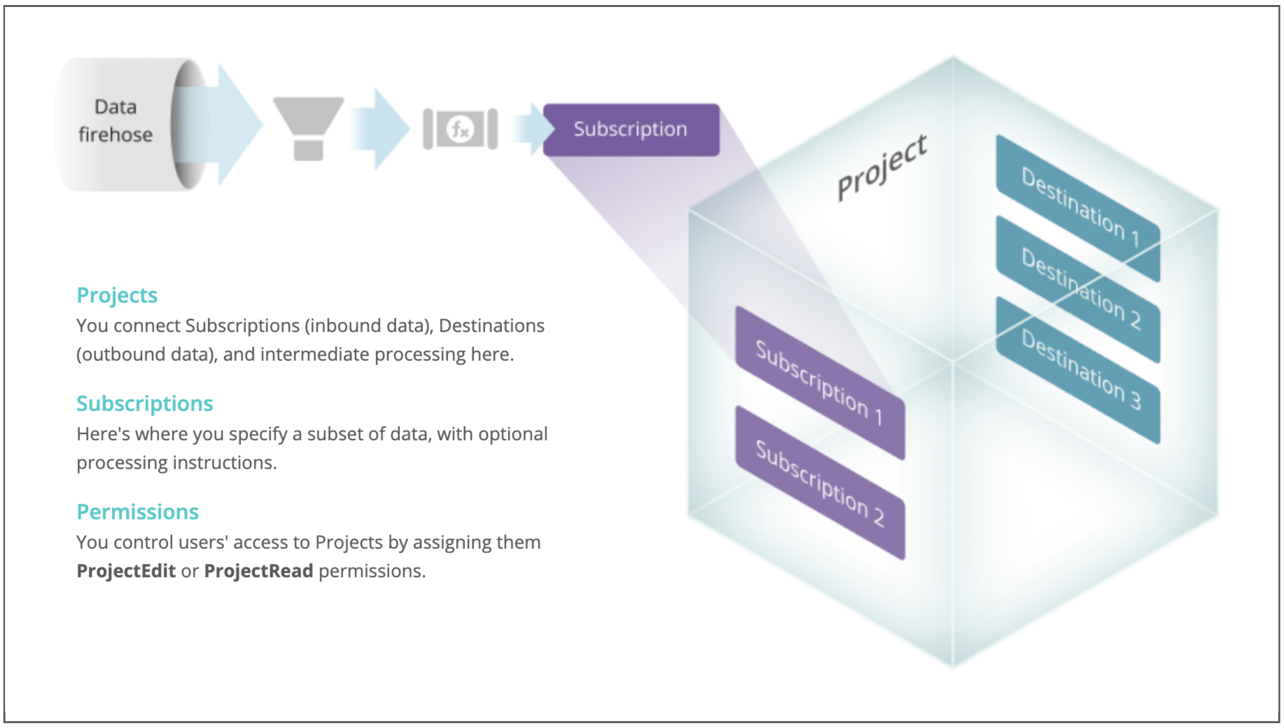
As a Stream Admin, you also control users' access to Projects and Subscriptions. For details, see Adding Users to Projects.

How the Components Fit Together

The components' names and relationship are easiest to remember in terms of the publish/subscribe model long used by Cribl Stream integrations like Kafka or Google Cloud Pub/Sub:

- A **Project** is a microcosm of a whole Cribl Stream deployment – gathering together Subscriptions' incoming data, routing/processing, Destinations, and authorized users.

- A **Subscription** specifies a subset of available data to retrieve, and how to import the data.


Data firehose
Subscription
Project
Destination 1
Destination 2
Destination 3
Subscription 1
Subscription 2
Projects
You connect Subscriptions (inbound data), Destinations (outbound data), and intermediate processing here.
Subscriptions
Here's where you specify a subset of data, with optional processing instructions.
Permissions
You control users' access to Projects by assigning them **ProjectEdit** or **ProjectRead** permissions.

How Subscriptions and Projects filter incoming data

Getting Started with Projects

You interact with Projects in either of two ways, depending on your Cribl Stream Permissions.

As a Project Editor

When you log into Cribl Stream and navigate to a Group, you'll find your Projects and Subscriptions already set up for you in a convenient visual UI.

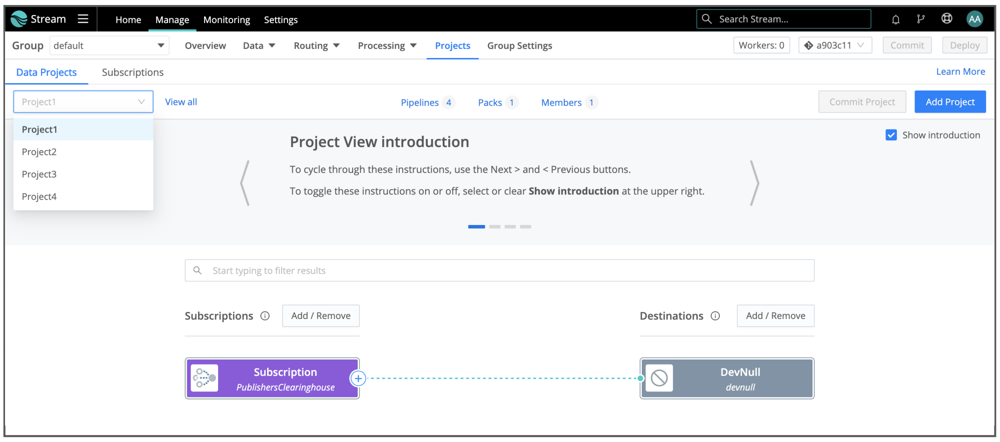
You can adjust your Projects' data flow by connecting Subscriptions to Destinations. You can insert and configure Pipelines, and can insert Packs, that are part of the Project. You can also commit your configuration changes to Git. This all works the same way as in QuickConnect.

As a Cribl Admin

To access Projects configuration options:

1. Navigate to a Worker Group. (Projects and their components are configured per Group, providing a first layer of isolation).
2. Select **Manage** > **Projects**. This exposes two lower tabs: **Data Projects** (the default, also called the Projects view) and **Subscriptions**.

3. See the following pages for details on configuring each of these components for Project editors.



Project View introduction

To cycle through these instructions, use the Next > and < Previous buttons.

To toggle these instructions on or off, select or clear **Show introduction** at the upper right.

Accessing the Project view

8.1. Configuring Projects

Projects define the data that the Project's users can consume, where that data can be sent, and who has access to the Project. A Project associates specific Subscriptions with specific Destinations. For fine-grained access control, you can share each Project with specific Members.

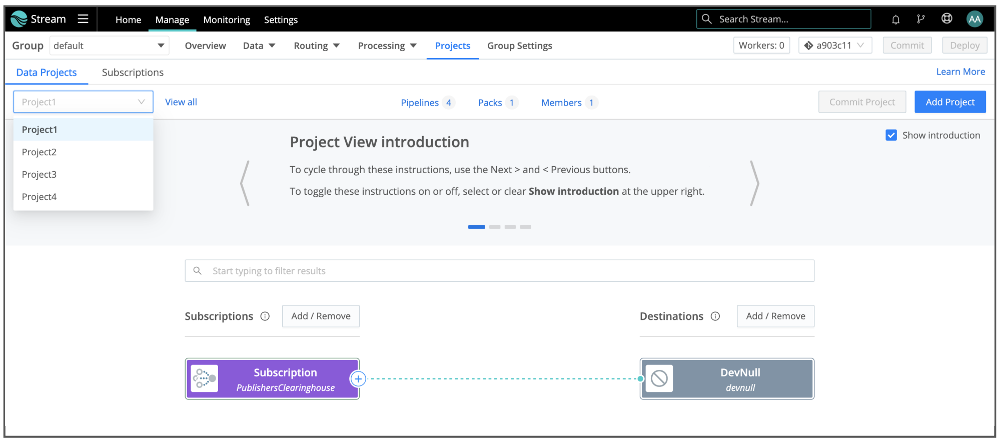
By connecting incoming data sources (Subscriptions) to specific Destinations, Projects function like a Route's Output selection. We expand on this similarity in Subscriptions and Routes.

Accessing Projects

From your Worker Group's **Manage** submenu, select **Projects** to open the default **Data Projects** lower tab.

This is the "Project view," an expanded version of the UI in which Project editors will manage connections among the resources you define here. From left to right, the options in this administrator's view are:

- Projects drop-down list: To bring a different Project into the Project view, select it here.
- **View all** link: Click to open the **All Projects** modal, where you can configure and share existing Projects.
- **Pipelines** link: Click to open a modal where you can manage Pipelines that are scoped to this Project.
- **Packs** link: Works the same way as the **Pipelines** link. The modal will show only Packs that are part of this Project. For Project editors, Packs will be read-only resources – Project editors can insert or remove Packs, but not modify them.
- **Commit Project** button: Commit changes from this Project to Git.
- **Add Project** button: Creates a new Project. Covered just below.



Project View introduction

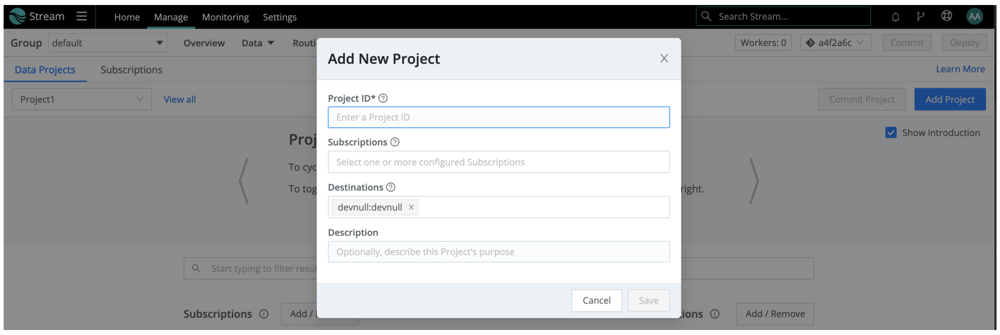
To cycle through these instructions, use the Next > and < Previous buttons.

To toggle these instructions on or off, select or clear **Show introduction** at the upper right.

Accessing the Project view

Configuring a New Project

1. From the Project view shown above, click **Add Project** to open the **Add New Project** modal shown below.



Configuring a new Project

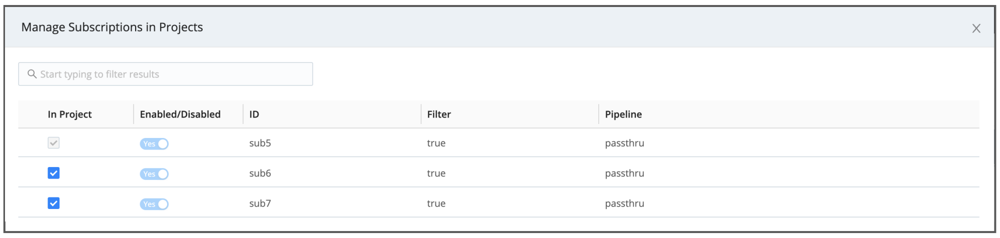
2. Assign this Project a unique **Project ID**. (This can include letters, numbers, `_`, and `-`. IDs are case-sensitive, so entering the same characters with different capitalization will create a separate ID.)
3. Select one or more configured Subscriptions. (You can access only Subscriptions that are already configured in this Project. But you can reopen this modal later to add newly configured Subscriptions. Also, the Subscriptions UI enables you to add Subscriptions to Projects.)
4. Select one or more configured Destinations that will have access to this Project's data. (You can access only Destinations that are already configured in this Project – but you can configure and add more Destinations in the next section.)
5. Optionally, add a **Description** of the Project's purpose.
6. Click **Save** to finish configuring your Project.
7. Commit and deploy your changes.

To view **Live Data** on your configured **Destination**, you must **Commit** and **Deploy** your changes.

Testing a Project (Managing Data Flow)

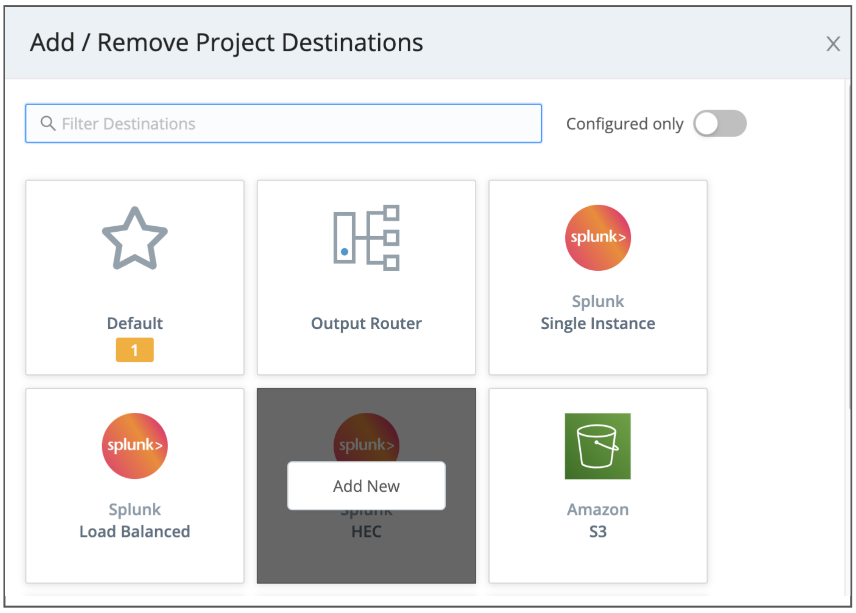
After you return to the Project view, make sure your desired Project has focus in the upper-left drop-down list. You can now use the Project view's visual UI to expand the Project and enable its data flow. The options here include:

- Connect the Project's already configured Subscriptions and Destinations, optionally adding a Pipeline or Pack to each connection.
- Click **Subscriptions** > **Add / Remove** to open a drawer where you can expose any Subscription you've already configured in the Project, remove Subscriptions from the Project view, or add new Subscriptions.


Manage Subscriptions in Projects

In Project	Enabled/Disabled	ID	Filter	Pipeline
✓	Yes	sub5	true	passthru
✓	Yes	sub6	true	passthru
✓	Yes	sub7	true	passthru

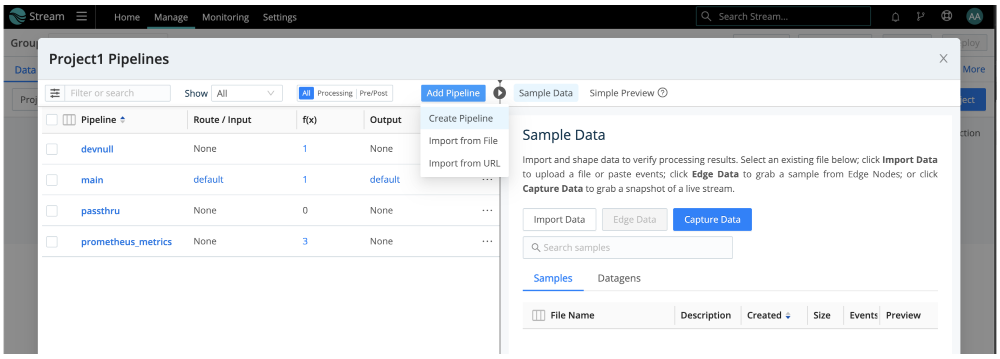
- Click **Destinations** > **Add / Remove** to open a drawer where you can expose and configure any Destination type available in your Cribl Stream deployment. (Any Destination that you add here will automatically be added to the Project's configuration modal when you next reopen it.)



As you configure and test your Project, beware of high ingestion rates that could overwhelm downstream receivers. To prevent dropped events, enable Persistent Queues on Destinations that support them.

Enabling Pipelines and Packs

Each Project that you configure opens with a basic set of Pipelines and one default Pack. To manage these, click the upper **Pipelines** or **Packs** link, respectively. Each opens a modal similar to Cribl Stream's standard Pipelines or Packs UI, with access to standard Sample Data and Preview options on the right.



Pipeline	Route / Input	f(x)	Output
devnull	None	1	None
main	default	1	default
passthru	None	0	None
prometheus_metrics	None	3	None

Sample Data

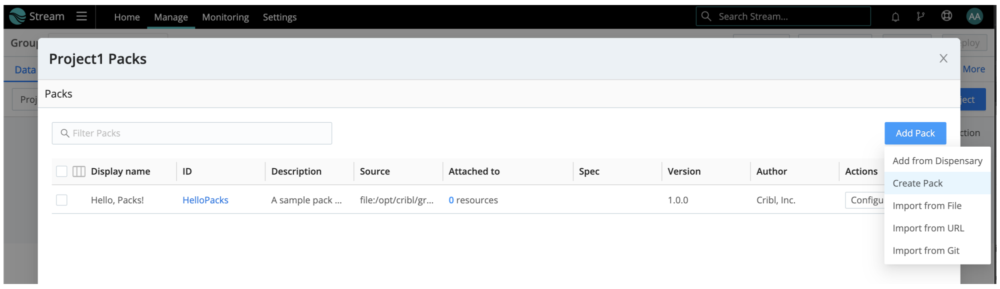
Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a snapshot of a live stream.

File Name	Description	Created	Size	Events	Preview

Managing Project Pipelines

To offer Project editors additional processing resources to insert on Subscription/Destination connections, click the modal's **Add Pipeline** or **Add Packs** button. You can then enable new Pipelines and Packs in the Project, with these constraints:

- You can create a new Pipeline or Pack here. It will be available only in the current Project.
- You can import an existing Pipeline or Pack. Any modifications that you make here will similarly be scoped to the current Project.
- To import an existing Pipeline from your Cribl Stream deployment, you must export it as JSON, then select the **Import from Pipeline** drop-down option shown above. (You can also **Import from URL**.)
- You can import Packs from an exported JSON file, a URL, or the Dispensary or Git options shown on the drop-down below.



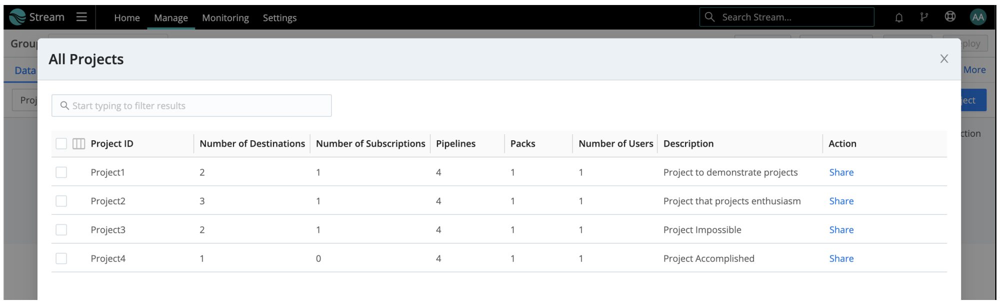
Display name	ID	Description	Source	Attached to	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack ...	file:/opt/cribl/gr...	0 resources		1.0.0	Cribl, Inc.	Configu

Adding Project Packs

Editing a Project

To update an existing Project, click the Project view's upper-left **View all** link. From the resulting **All Projects** modal, click the Project's row to reopen its configuration in a modal similar to the **Add New Project** modal covered above.

You can also click any row's **Share** link to open a drawer in which to manage users' access to the Project.



Project ID	Number of Destinations	Number of Subscriptions	Pipelines	Packs	Number of Users	Description	Action
Project1	2	1	4	1	1	Project to demonstrate projects	Share
Project2	3	1	4	1	1	Project that projects enthusiasm	Share
Project3	2	1	4	1	1	Project Impossible	Share
Project4	1	0	4	1	1	Project Accomplished	Share

View all > All Projects modal

Accessing Project Metrics

You can access the following metrics on your Projects:

- Events and bytes in/out of Subscriptions.
- Bytes in/out of Pipelines, Packs, and Destinations.
- Events in/out/dropped through Pipelines, Packs, and Destinations;

To tap these metrics, enable Cribl Stream's internal CriblMetrics Source. This Source's defaults are different for Cribl.Cloud versus on-prem deployments.

Cribl.Cloud Leaders and Groups

For Cribl.Cloud Leaders, and Worker Groups of Cribl-managed Cribl.Cloud Workers, Projects' metrics are always available to the CriblMetrics Source. This availability is not configurable.

On-Prem Leaders and Groups

For on-prem Leaders, on-prem Worker Groups, and hybrid Worker Groups, Projects' metrics are disabled by default. (As with other on-prem exclusions, we provide this default to optimize performance.) However, you can easily make these metrics available to the CriblMetrics Source.

See Controlling Metrics for general instructions on modifying defaults at the Leader, Worker Group, or single-instance level. From the **Disable field metrics** list, remove the `project` metric. Then save, commit, and deploy your changes.

8.2. Configuring Subscriptions

You use a Subscription's **Filter** expression to specify a subset of a Worker Group's overall incoming data stream to forward to Destinations. Optionally, you can also specify a Pre-processing Pipeline to condition that data before it leaves the Subscription.

You gather Subscriptions into Projects, where you match them up with Destinations and share them with authorized users.

A Project can contain multiple Subscriptions. In Cribl Stream 4.2 and later, each Subscription can be associated with multiple Projects. Each Project's copy of the Subscription will share the same configuration, but will have independent data flow.

Subscriptions and Routes

From the UI similarity, you've probably noticed that Subscriptions look and function much like Routes. There are both similarities and differences – which we'll consider in reverse order.

Subscriptions/Routes Differences

Compared to Routes, Subscriptions present these exceptions:

- Subscriptions lack a Route's **Output** or **Output Expression** field. (Instead, you associate Subscriptions with Projects, where Project editors can process the data and send it to Destinations.
- Subscriptions have no **Final** flag. So a Subscription is similar to a Route with the **Final** toggle switched off.
- Subscriptions do not relay backpressure upstream to data senders. (This is by design, to prevent any Subscription from affecting data flow through parallel Subscriptions, Projects, or Routes/Pipelines.) If any backpressure occurs, a Subscription will drop data.

Subscriptions/Routes Similarities

A further link to Cribl Stream's Data Routes interface is that Subscriptions can ingest data **only** from Sources configured (at the Cribl Stream product level) as **Send to Routes**. (Subscriptions and Projects cannot ingest data from QuickConnect Sources, despite the similarity of Projects' visual UI (the Project view).

This also means that Subscriptions tap exactly the same overall flow of inbound data that Data Routes do, in parallel. So there are cases where, after you've configured Subscriptions to ingest certain data, you might want to configure a Pipeline at the top of the Routing table to drop the same data. This will prevent Cribl Stream from processing the same events twice.

Usage Tips

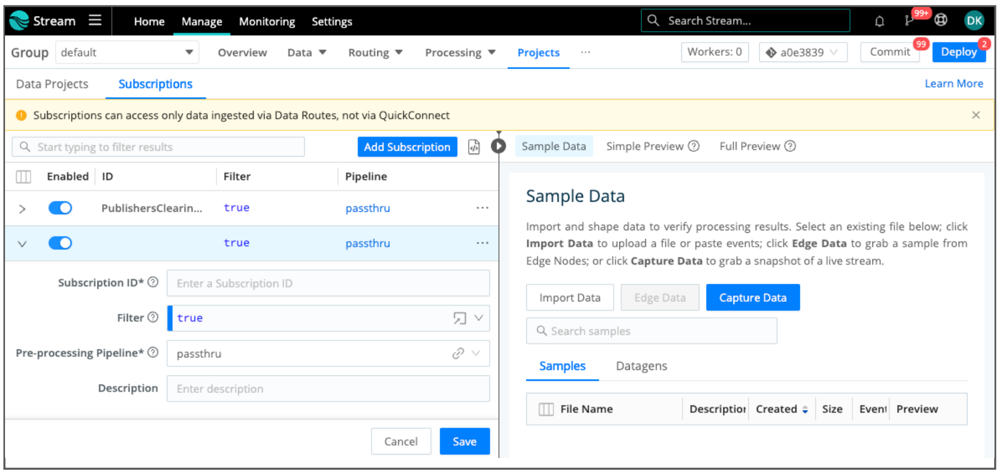
Compared to sending the same data flow through Routes, adding Subscriptions can have a performance impact, because each Subscription creates a copy of the data that Cribl Stream needs to process. Therefore, your resource requirements could change with the number of Subscriptions you add.

Please work with your Cribl Solutions Engineer to select an architecture that best balances your needs for performance versus segmented data/configuration access.

Cribl recommends that you configure a separate Subscription – and corresponding Pipeline(s) – for each data type you ingest. This enables you to filter by data type in the Subscription's definition, and again in individual Pipeline Functions.

Accessing Subscriptions

1. From your Worker Group's **Manage** submenu, select **Projects**.
2. Select the **Subscriptions** lower tab.
   Any Subscriptions you've already configured will appear here. Expand the accordions to access each Subscription's configuration.



Subscriptions can access only data ingested via Data Routes, not via QuickConnect

Enabled	ID	Filter	Pipeline
●	PublishersClearin...	true	passthru
●		true	passthru

Subscription ID*: Enter a Subscription ID
Filter: true
Pre-processing Pipeline*: passthru
Description: Enter description

Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a snapshot of a live stream.

Configuring a new Subscription

Configuring a New Subscription

1. From the **Subscriptions** lower tab, click **Add Subscription** to open the configuration options described below.
2. Assign this Subscription a unique **Subscription ID**. (This can include letters, numbers, `_`, and `-`. IDs are case-sensitive, so entering the same characters with different capitalization will create a separate ID.)

3. Modify the **Filter** by replacing the default `true` with a JavaScript expression that selects a subset of events to filter through this Subscription. (Your expression can specify an upstream sender, a data type, the presence of particular fields, etc. For examples, see Cribl Expression Syntax.)
4. Modify (as needed) the Source's default `passthru` Pre-processing Pipeline.
5. Optionally, add a **Description** of the Subscription's purpose.
6. Optionally, assign the Subscription to one or more Projects. (You can also import Subscriptions from individual Projects' configurations, and can control Subscriptions' visibility in the Project view.)
7. Click **Save** to save the new Subscription to the Subscriptions table.
8. Commit and deploy your changes.

A Subscription must specify a Pre-processing Pipeline, even if this is the default `passthtru`, which performs no processing.

Adding and Previewing More Subscriptions

From the **Subscriptions** tab, you can click **Add Subscription** repeatedly to define multiple Subscriptions. These stack on the left, like Routes in Cribl Stream's Routing table.

Note these further similarities to the Routing table:

- Each Subscription's blue title bar has a toggle that you can use to disable the Subscription from gathering data, while preserving its configuration.
- On the right, you can capture, import, and save sample data, and. You can use these samples to preview how your Subscriptions transform incoming data.

Editing a Subscription

To update an existing Subscription, just click its accordion in the Subscriptions table. This will reopen the same configuration options outlined above.

8.3. Adding Users To Projects

As a Stream admin, you control Members' access to individual Projects by assigning Permissions to those Members, as outlined below.

Breaking Change

In Cribl Stream 4.2 and later, Projects and Subscriptions rely entirely on the fine-grained Members/Permissions access control first introduced in Stream 4.2. The 4.1.x `project_user` Role and `ProjectSourceSubscribe` Policy, which provided access to all Projects, are now retired. To migrate Projects configured in v.4.1.x, you'll find those projects' users visible in Stream as Members. Assign them appropriate new Permissions, as outlined here.

Project-Level Permissions

You can share a Project with a Cribl Stream Member using the following Permissions, with some constraints inherited from the Member's Permissions at higher levels (Cribl Organization, Cribl Stream, and Worker Group).

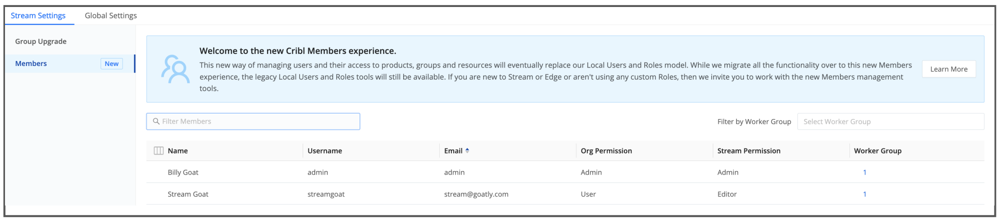
Permission	Description
Maintainer	Admin-level Permission. Allows freely editing or deleting the Project and its settings. (Shared, by inheritance, only with Members who have the `Admin` or `Editor` Permission at higher levels.)
Editor	Can configure connections among the Project's Subscriptions, Packs, and Destinations, but cannot modify or delete these resources. Can create, modify, and delete Pipelines within the Project. (**Use this Permission to share a Project with** Project editors.)
Read Only	Can view Project and Subscription settings and connections, but not modify or delete them. (Designed specifically for support personnel. Members who have the product-level `Read Only` Permission will inherit, and will be locked to, this Permission.)
No Access	Self-explanatory: a Member's default state on a Project that has not (yet) been shared with them.

Members who have the `User` Permission at the top (Organization) level are the most malleable at the Project level. These are the Members with whom you can readily share a Project-specific `Editor` Permission, to authorize them as a Project editor.

Sharing a Project (Example)

To share a Project with an eligible Cribl Stream Member:

1. Select **Settings** > **Stream Settings** > **Members**.
2. Make sure each applicable Member has the `User` **Org Permission**, and the `User` **Stream Permission**. (Click an existing Member's row to adjust their Permissions.)

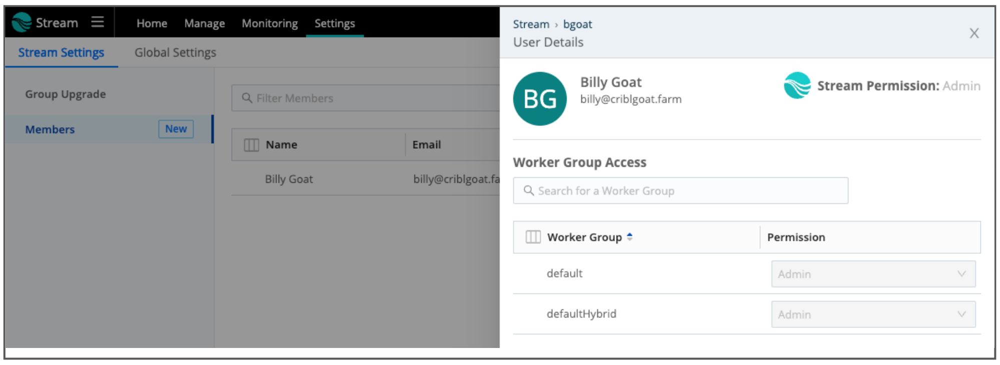


Welcome to the new Cribl Members experience.
This new way of managing users and their access to products, groups and resources will eventually replace our Local Users and Roles model. While we migrate all the functionality over to this new Members experience, the legacy Local Users and Roles tools will still be available. If you are new to Stream or Edge or aren't using any custom Roles, then we invite you to work with the new Members management tools.

Name	Username	Email	Org Permission	Stream Permission	Worker Group
Billy Goat	admin	admin	Admin	Admin	1
Stream Goat	streamgoat	stream@goatly.com	User	Editor	1

Stream Settings > Members

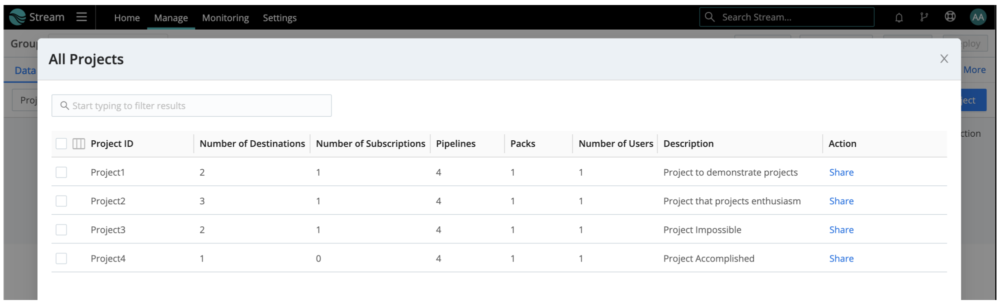
3. Click each applicable Member's row to open their **User Details** drawer.



Worker Group	Permission
default	Admin
defaultHybrid	Admin

Stream Members > User Details

4. Grant the Member a `User` Permission on the applicable Project's parent Group. (Higher Permissions offered on this drop-down are not applicable to Project editors.)
5. Select **Manage** and click the same Group to give it focus.
6. Select **Projects** > **Data Projects** > **View all**.



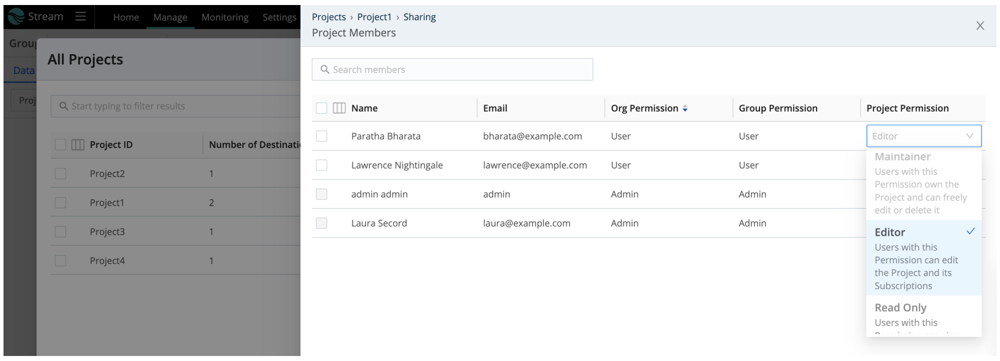
Project ID	Number of Destinations	Number of Subscriptions	Pipelines	Packs	Number of Users	Description	Action
Project1	2	1	4	1	1	Project to demonstrate projects	Share
Project2	3	1	4	1	1	Project that projects enthusiasm	Share
Project3	2	1	4	1	1	Project Impossible	Share
Project4	1	0	4	1	1	Project Accomplished	Share

View all > All Projects modal

7. In the resulting **All Projects** modal, click the Project's **Share** link to open the **Project Members** drawer. Here, you can see all the Group's Members with whom this Project can be (or has been) shared.

In Cribl Stream 4.3 and later, you can instead open an individual Project's **Sharing** drawer directly from the Project view. Give this Project focus in the Project view, then click the **Members** link at the top.

8. Grant each applicable Member an `Editor` Permission on the Project. (The alternative `Read Only` Permission is intended for support users. It does not enable configuring connections or Pipelines.)



Project ID	Number of Destinati
Project2	1
Project1	2
Project3	1
Project4	1

Name	Email	Org Permission	Group Permission	Project Permission
Paratha Bharata	bharata@example.com	User	User	Editor
Lawrence Nightingale	lawrence@example.com	User	User	
admin admin	admin	Admin	Admin	
Laura Secord	laura@example.com	Admin	Admin	

Maintainer
Users with this Permission own the Project and can freely edit or delete it

Editor ✓
Users with this Permission can edit the Project and its Subscriptions

Read Only
Users with this

Project Sharing drawer – assigning `Editor` Permission

9. Click **Save** to confirm your changes, then click outside the drawer to close it.
10. Commit your changes, and deploy the Group if prompted.

These Members who now have Project `Editor` Permissions – and no higher Stream Permissions – view data filtered through the Projects UI. They also have read access to Cribl Stream Data Monitoring and Notifications.

You can reopen the same drawer to modify Members' Permission levels. To remove a Member from the Project, assign them `No Access`.

9. Administering

9.1. Licensing

Every Cribl Stream 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. For basic information about Cribl.Cloud Enterprise and Standard plans, see Cloud Pricing.

Managing Licenses – Adding and Renewing

On the Leader (or in a single-instance deployment), after submitting your newly untarred Cribl Stream software's registration page, 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 Stream 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.

Airgapped Deployments

If your environment does not allow internet connectivity, you'll need to bypass the registration Web form when you deploy or upgrade Cribl Stream. To do so:

1. On the Leader's (or single instance's) host, navigate to `$CRIBL_HOME/local/cribl`.
2. Create a file named `license.yml`.
3. Paste in your license key in this format, then save the result:

```
licenses:
- <your-license-key-here>
```

Expiration/Renewal Notifications

In Cribl Stream 4.3 and later, the UI will prompt you to renew your paid license by displaying banner and drawer notifications, starting 30 days before license expiration. When you see these prompts, verify your license's expiration date and contact Cribl Sales at sales@cribl.io to renew.

Beyond these default notifications, you can configure custom License-Expiration Notifications. You can send these to PagerDuty, or to other external services via webhook.

Exemptions from License Quotas

Cribl does not require a separate license for sending data from Cribl Stream to Cribl Stream, 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.
- Dropped events do not count against your ingestion quota.

License Types

Cribl offers several Cribl Stream license types including Enterprise, Standard, and Free. For a current list of available license types and a detailed comparison of what's included in each, see Cribl Pricing.

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.

License Expiration

When an Enterprise or Standard license expires, you can remove the old license to fall back to the Sales Trial or Free type. An expired Sales Trial license cannot fall back to a Free license.

To delete your expired license:

1. Select **Settings** then **Licensing**.
2. Locate your expired license, and click **Delete license**.
3. Restart the Leader.

> Upon expiration of a paid license, if there is no fallback license, Cribl Stream will backpressure and block all incoming data.

Licensing in Distributed Deployments

With licenses that limit the number of Worker Processes/Edge Nodes, Cribl Stream will attempt to balance or rebalance Worker Processes (threads) as evenly as possible across all licensed Worker Nodes.

You configure licensing only on the Leader Node. (See Managing Licenses – Adding and Renewing.) The Leader will push license information down to Worker Groups as part of the heartbeat (Stream, Edge). There is no need to configure or store licenses directly on Worker Nodes.

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 Stream 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 Stream. With a Free license, disabling telemetry will cause Cribl Stream 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 Stream's UI at **Settings** > **Global Settings** > **System** > **General Settings** > **Upgrade & Share Settings** > **Share telemetry with Cribl**. Set the toggle to `No`.

Metadata Shared Through Telemetry

Your Cribl Stream instance shares metadata with Cribl per interval (roughly, every minute).

Details sent for nodes in any mode:

- Version
- OS Distribution/Version
- Kernel Version
- Instance's GUID
- License ID
- Earliest, Latest Time
- Number of Events In and Out, overall and by Source type and Destination type
- Number of Bytes In and Out, overall and by Source type and Destination type
- Number of Open, Closed, Active Connections
- Number of Routes
- Number of Pipelines
- Runtime Environment Indicators (AWS and Kubernetes)

Additional details sent for Leader nodes (when Cribl isn't sending managed node data):

- Runtime Environment
- OS Distribution / Version

- Version
- Kernel Version

How We Use Telemetry Data

With telemetry enabled, Cribl software sends usage data directly to Cribl. We securely store and encrypt this data on Cribl-managed servers, with restricted access.

Cribl fully anonymizes and aggregates this usage data in our systems of analysis. There, we use the aggregated metrics to improve Cribl products and services by analyzing circumstances around disruptions, opportunities for ingest efficiencies, and ways to optimize performance. Access is restricted to only those Cribl employees and contractors who require anonymized data to perform their jobs.

Cribl collects License IDs only to ensure that all data is being sent by a Cribl product. These IDs cannot be used to personally identify any user of Cribl software or Cribl cloud services.

For further details, see the Cribl Privacy Policy.

Licensing FAQ

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

Open Cribl Stream'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:

- 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 Stream 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 Stream 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 Stream 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.

9.2. Version Control

Tracking, backing up, and restoring configuration changes for single-instance and distributed deployments

---

Cribl Stream integrates with Git clients and remote repositories to provide version control of Cribl Stream's configuration. This integration offers backup and rollback for single-instance and distributed deployments.

These options are separate from the Git repo responsible for version control of Worker configurations, located on the Leader Node in distributed deployments. We cover all these options and requirements below.

> Cribl.Cloud deployments do not currently support integration with external Git clients or remote repos.

Git Installation (Local or Standalone/Single-Instance)

To verify that `git` is available, run:

```
git --version
```

The minimum version that Cribl Stream requires is: **1.8.3.1.** If you don't have `git` installed, see the installation links here.

Git Required for Some Features

Git is a hard requirement for certain Cribl Stream features.

Distributed Deployments

For distributed deployments, `git` **must** be installed and available locally on the host running the Leader Node.

All configuration changes must be committed before they are deployed. The Leader notifies Workers that a new configuration is available, and Workers pull the new configuration from the Leader Node.

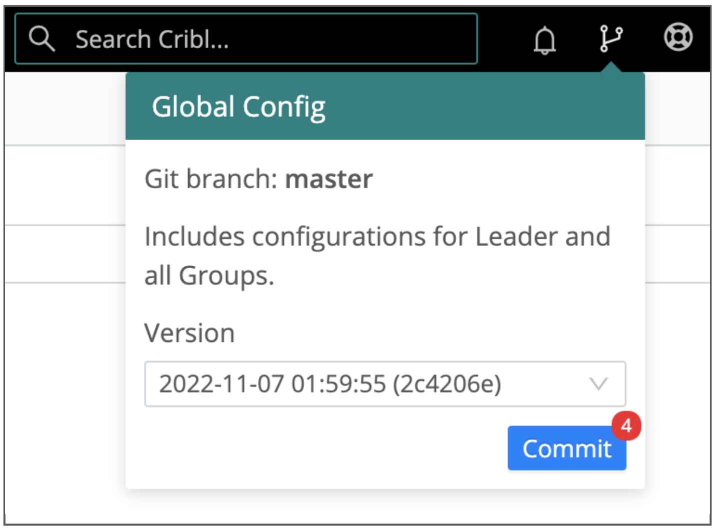
Licensing Dashboard

Even on single-instance deployments, the Monitoring > Licensing dashboard will display configuration change markers only if you have `git` installed.

Committing Changes

Once Git is installed, you can commit configuration changes using the `git` CLI. You can also commit changes interactively, using Cribl Stream's UI.

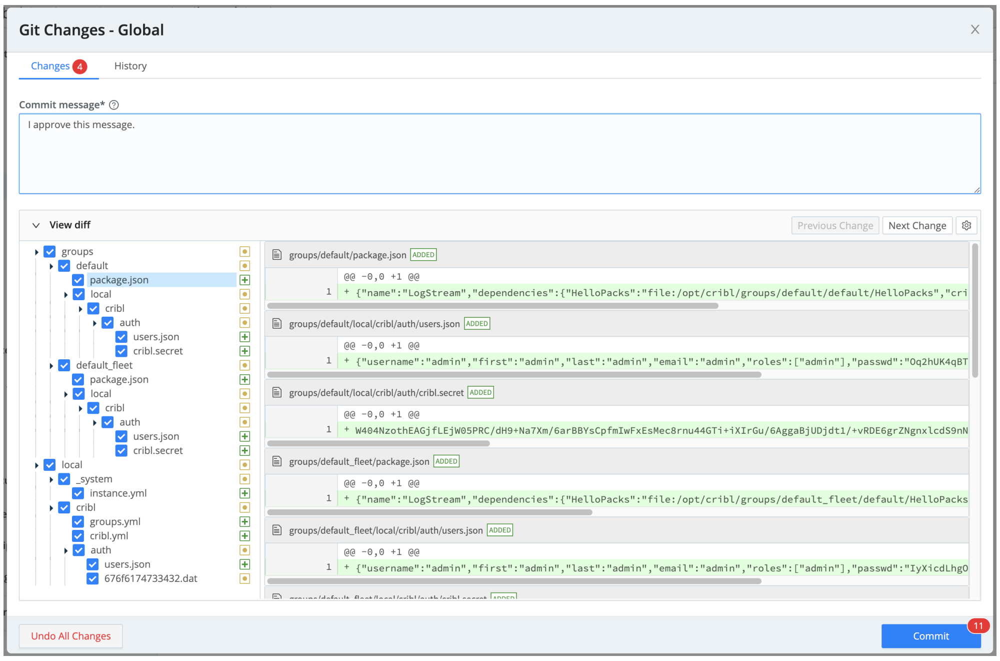
Pending commits have a red dot indicator, as shown below. Click **Commit** to proceed.



Search Cribl...
Global Config
Git branch: master
Includes configurations for Leader and all Groups.
Version
2022-11-07 01:59:55 (2c4206e)
Commit
4

Changes pending commit

Next, in the resulting **Commit Changes** modal, you can verify the diff'ed configuration changes. Other options here include clearing individual files' check boxes to exclude them from the commit (as shown below), and clicking **Undo** to reverse the changes instead of committing them.



Commit message*

I approve this message.

groups
default
package.json
local
cribl
auth
users.json
cribl.secret
default_fleet
package.json
local
cribl
auth
users.json
cribl.secret
local
_system
instance.yml
cribl
groups.yml
cribl.yml
auth
users.json
676f6174733432.dat

```
groups/default/package.json ADDED
     @@ -0,0 +1 @@
   1 + {"name":"LogStream","dependencies":{"HelloPacks":"file:/opt/cribl/groups/default/default/HelloPacks","cri
groups/default/local/cribl/auth/users.json ADDED
     @@ -0,0 +1 @@
   1 + {"username":"admin","first":"admin","last":"admin","email":"admin","roles":["admin"],"passwd":"Oq2hUK4qBT
groups/default/local/cribl/auth/cribl.secret ADDED
     @@ -0,0 +1 @@
   1 + W404NzothEAGjfLEjW05PRC/dH9+Na7Xm/6arBBYsCpfmIwFxEsMec8rnu44GTi+iXIrGu/6AggaBjUDjdt1/+vRDE6grZNgnxlcdS9nN
groups/default_fleet/package.json ADDED
     @@ -0,0 +1 @@
   1 + {"name":"LogStream","dependencies":{"HelloPacks":"file:/opt/cribl/groups/default_fleet/default/HelloPacks
groups/default_fleet/local/cribl/auth/users.json ADDED
     @@ -0,0 +1 @@
   1 + {"username":"admin","first":"admin","last":"admin","email":"admin","roles":["admin"],"passwd":"IyXicdLhgO
```

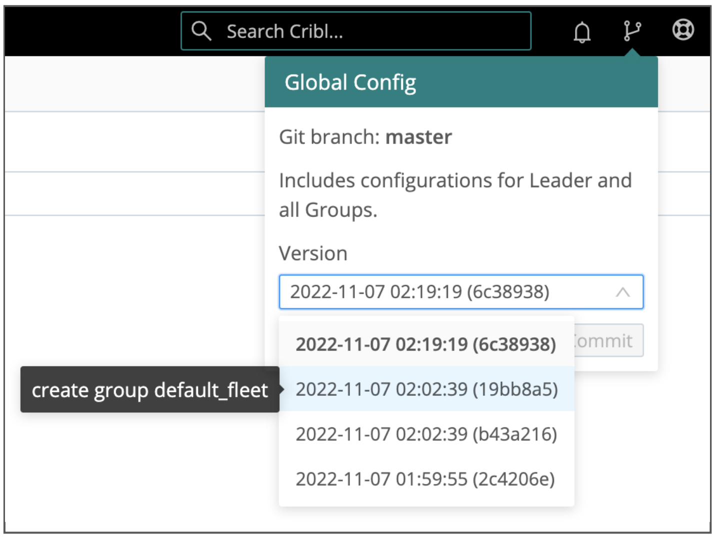
Reviewing a pending commit

When you’re ready to commit to your commit, click **Commit**. Look for a **Commit successful** confirmation message at the lower right.

Reverting Commits

Once Git is installed, you can revert to a previous commit using the `git` CLI. You can also restore a Worker Group’s previous commit using Cribl Stream’s UI:

Select the commit from the **Config Version** drop-down, as shown below. (Note that you’re selecting not a single commit to roll back, but the commit point to roll back **to**.)


Global Config

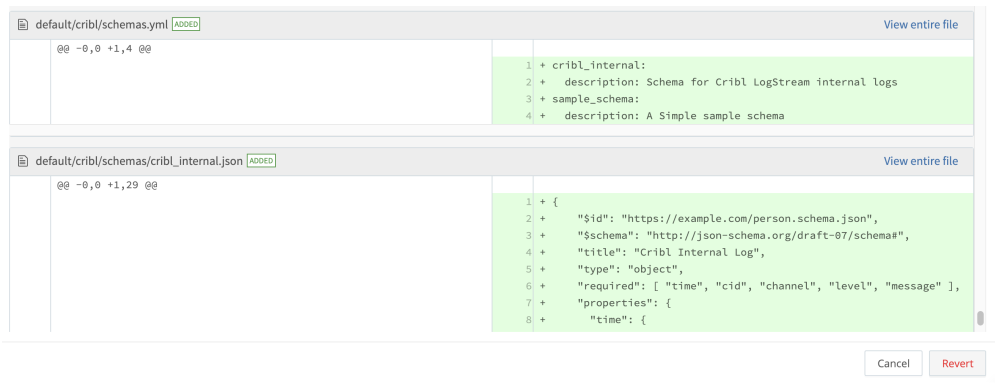
Git branch: **master**

Includes configurations for Leader and all Groups.

Version: 2022-11-07 02:19:19 (6c38938)

- **2022-11-07 02:19:19 (6c38938)**
- 2022-11-07 02:02:39 (19bb8a5) — create group default_fleet
- 2022-11-07 02:02:39 (b43a216)
- 2022-11-07 01:59:55 (2c4206e)

Then, in the resulting **Commit** modal, verify the diff’ed configuration changes and click **Revert**.


default/cribl/schemas.yml ADDED

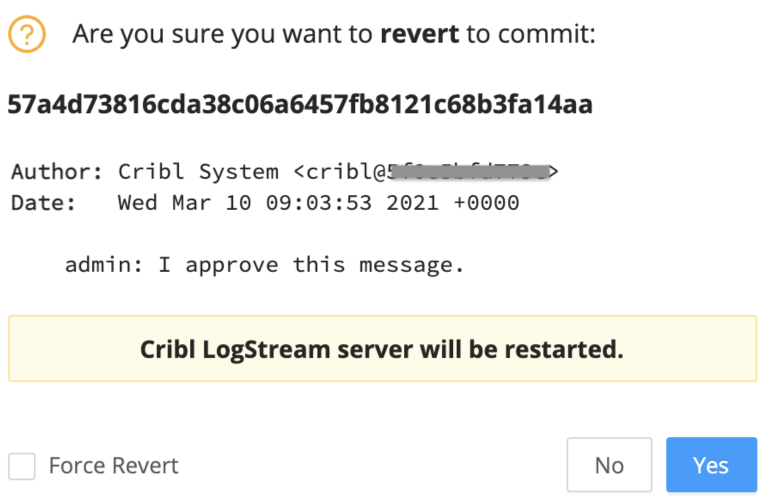
```
@@ -0,0 +1,4 @@
+ cribl_internal:
+   description: Schema for Cribl LogStream internal logs
+ sample_schema:
+   description: A Simple sample schema
```

default/cribl/schemas/cribl_internal.json ADDED

```
@@ -0,0 +1,29 @@
+ {
+     "$id": "https://example.com/person.schema.json",
+     "$schema": "http://json-schema.org/draft-07/schema#",
+     "title": "Cribl Internal Log",
+     "type": "object",
+     "required": [ "time", "cid", "channel", "level", "message" ],
+     "properties": {
+       "time": {
```

Undoing earlier commits

Finally, confirm permission for Cribl Stream to restart.


Are you sure you want to **revert** to commit:

57a4d73816cda38c06a6457fb8121c68b3fa14aa

```
Author: Cribl System <cribl@[redacted]>
Date:   Wed Mar 10 09:03:53 2021 +0000

    admin: I approve this message.
```

Cribl LogStream server will be restarted.

Support For Remote Repositories

Git **remote** repositories are supported – but not required – for version control of all configuration changes.

This feature requires a Cribl Stream Enterprise or Standard license.

You can configure a Leader Node with Git remote push capabilities through the Cribl Stream CLI, or through the Cribl Stream UI (via **Settings** > **Global Settings** > **Distributed Settings > Git Settings**).

To create a repo, see these tutorials:

- Setting Up a Repository (CLI instructions, host-agnostic, from Atlassian).
- Creating a New Repository (specific to GitHub's Web UI).
- Create a Repo (longer GitHub-specific tutorial, also covers committing changes).

Remote Formats Supported

Remote URI schema patterns should match this regex: `(?:git|ssh|ftps?|file|https?|git@[-\w.]+):(\/\/)?(.*?)(\.git\/?)?$`.

You can find a list of supported formats here.

For example:

- GitHub or other providers: `<protocol>://git@example.com/<username>/<reponame>.git`
- Local Git servers: `git://<host.xyz>:<port>/<user>/path/to/repo.git`

Securing Local Configuration and Remote Repos

Make a local backup of your whole `$CRIBL_HOME` directory before you set a remote repo. This safeguards your Cribl Stream configuration against being overwritten by a misconfigured remote.

Create each new remote as an **empty** repo, with no readme file and no commit history. (See the GitHub and GitLab examples below.)

Some files that are used by Cribl Stream (both Leader and Worker Groups) contain sensitive keys; examples are `cribl.secret` and `...auth/ssh/git.key`. These will be pushed to the remote repo as part of the entire directory structure under version control. Ensure that this repo is secured appropriately.

Connecting to a Remote Repo over HTTPS (Recommended)

Cribl recommends connecting to a remote repo over HTTPS. The examples below show a token-based and a password-based HTTPS connection to GitHub.

Example: Connecting to GitHub over HTTPS with a Personal Access Token

1. Create a new GitHub repository.
   For best results, create a new **empty** repo, with no readme file and no commit history. (This will prevent `git push` errors.) Note the username and email associated with your login on the repo provider.
2. Create a personal access token with **repo** scope.
3. Copy the token to your clipboard.
4. In Cribl Stream, go to **Settings** > **Global Settings** > **System** > **Git Settings**.
5. In the **Remote** tab, select **Authentication type**: **None**.
6. Fill in the **Remote URL** field with your username and repo name. Use the format below:
   `https://<username>:<token>@github.com/<owner>/<repository>.git`

For additional details, see GitHub's Creating a Personal Access Token tutorial.

For GitHub repos specifically, use only personal access tokens in the **Remote URL** field. GitHub announced its end of support for plaintext passwords as of August 13, 2021.

Example: Connecting to GitHub over HTTPS with Username and Password

1. Create a new GitHub repository.
   For best results, create a new **empty** repo, with no readme file and no commit history. (This will prevent `git push` errors.) Note the username and email associated with your login on the repo provider.
2. In Cribl Stream, go to **Settings** > **Global Settings** > **System** > **Git Settings**.
3. In the **Remote** tab, select **Authentication type**: **Basic**.
4. Enter your credentials in the **User** and **Password** fields.

Connecting to a Remote with SSH

You can set up SSH keys from the CLI, or upload keys via the UI. If you have a passphrase set, this functionality is available only through the CLI – see Encryption: Configuring Keys with the CLI. The example below outlines the UI steps.

Example: Connecting to GitHub with SSH

1. Create a new GitHub repository.
   For best results, create a new **empty** repo, with no readme file and no commit history. (This will prevent `git push` errors.) Note the user name and email associated with your login to the repo provider.
2. Add an SSH public key to your GitHub account.
3. In Cribl Stream, go to global **Settings** > **Global Settings** > **System** > **Git Settings** > **Remote**.
4. Fill in the **Remote [repo] URL**. In the generic example below, replace `<username>` with your user name on the repo provider:
   Remote URL: `<protocol>://git@github.com:<username>/<reponame>.git`

For GitHub specifically, the URL/protocol format must be:
Remote URL: `git@github.com:<user>/<reponame>.git`

A specific (fictitious) GitHub example:
Remote URL: `git@github.com:taylorswift/leadsheets.git`

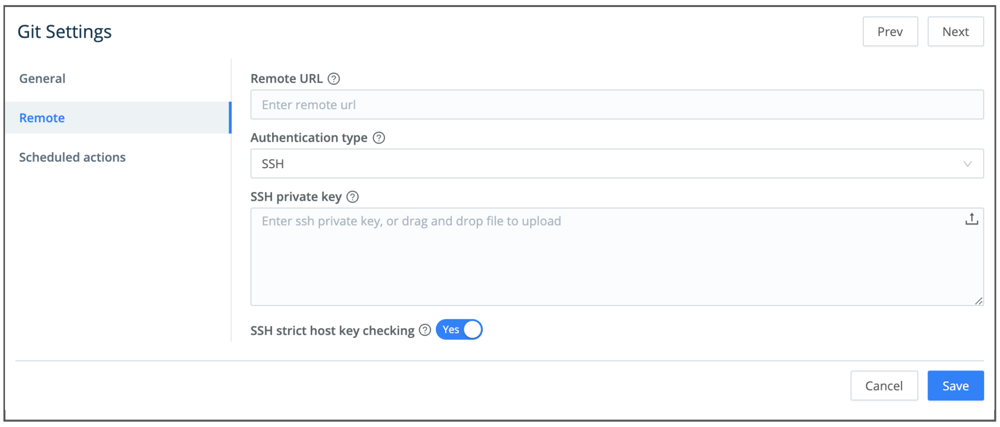
5. Paste in the **SSH private key**.

The key will paste in with an appended newline below it. Do not delete this newline before you save the Remote Settings, or else you will trigger an error of the form: `fatal: Could not read from remote repository.`

6. As the user running Cribl Stream, run this command to add the GitHub keys to `known_hosts`:

```
ssh-keyscan -H github.com >> ~/.ssh/known_hosts
```

For additional details, see GitHub's Connecting to GitHub with SSH tutorial.


Git Settings

Remote URL: Enter remote url

Authentication type: SSH

SSH private key: Enter ssh private key, or drag and drop file to upload

SSH strict host key checking: Yes

Cribl Stream's Git settings

GitLab Notes

For repos hosted on GitLab, Cribl's general recommendations are:

- Create a new empty (blank) project, with no readme file and no commit history. (Clear the **Initialize repository with a README** check box.)

  This will prevent `git push` errors by creating a project with no branch, not even `main`. When you later push Cribl Stream's `master` branch, that will create a corresponding `master` branch on the GitLab remote with no conflicts.

- Create a GitLab project access token for authentication. See GitLab's documentation, which also covers **project bot** conventions.
- With project bots, the first token's username is set to `project_{project_id}_bot`. The password is the alphanumeric token.
- Create the token with `write_repository` scope.
- Specify a remote URL in HTTPS format – e.g.: `https://localgitlab.<yourdomain.ext>/<yourusername>/cribl.git`

- If you do see a `main` branch in Cribl Stream's UI, do not select it. Leave the branch at `master`, then use the Stream UI's **Commit** and **Push** commands. This will create the desired `master` branch on GitLab's remote.

GitLab's **Repository Settings** > **Push Rules** section includes these two settings of interest:

- As needed, enable **Check whether author is a GitLab user**.
- Understand the consequences of of enabling **Prevent committing secrets to Git**. This blocks commits of `.pem` and `.key` files. If you have certificates or SSH keys configured, this will break commits from Cribl Stream, throwing only a generic `API Error` in the UI. Check your `git` CLI client for more-specific diagnostics.

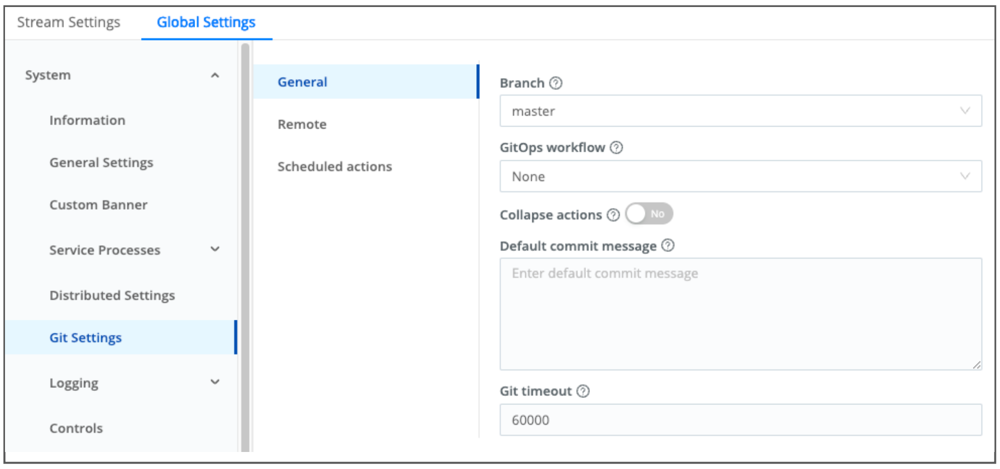
Additional Git Settings

On the **Git Settings** > **General** tab, you can modify the following configurations, all of which are optional.

- **Branch**, **GitOps workflow**: These drop-downs are available to configure GitOps.

  See the GitOps documentation linked above for details and precautions.

- **Collapse actions**: Combine multiple Git buttons to one, to reduce repetitive clicks. See Collapse Actions below.
- **Default commit message**: Enter a placeholder message to apply to all commits from Cribl Stream. This also reduces clicks.
- **Git timeout**: Maximum time (in milliseconds) to wait for Git processes to complete before killing them. If a Leader instance hangs, try lowering this value from the default `60000` (60 seconds). However, avoid very low values (like `20` ms) – which will time out even when Git processes are working properly. Enter `0` to wait indefinitely.

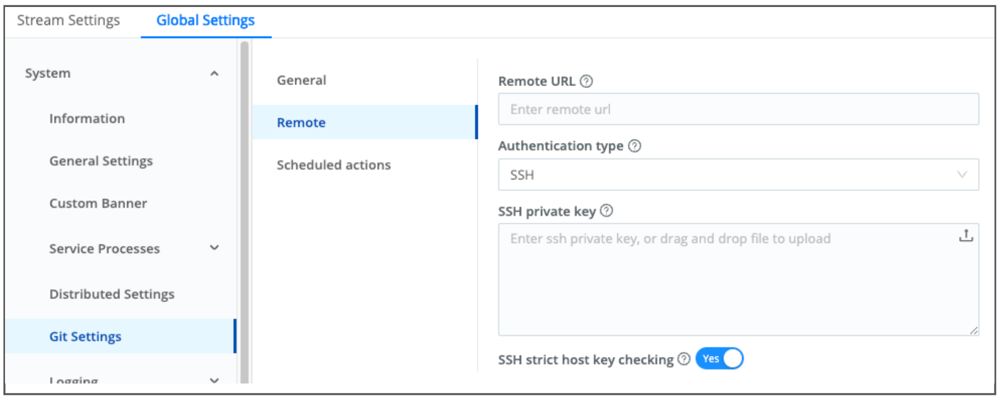


Branch: master
GitOps workflow: None
Collapse actions: No
Default commit message: Enter default commit message
Git timeout: 60000

Git Authentication Type settings

On the **Git Settings** > **Remote** tab, you can change the **Authentication Type** from its **SSH** default to **Basic** authentication. This displays two additional fields:

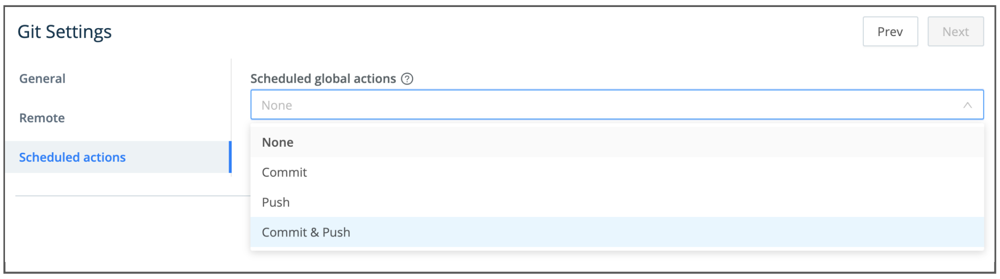
- **User**: Username on the repo.
- **Password**: Authentication password (e.g., a GitHub personal access token).



Git Authentication Type settings

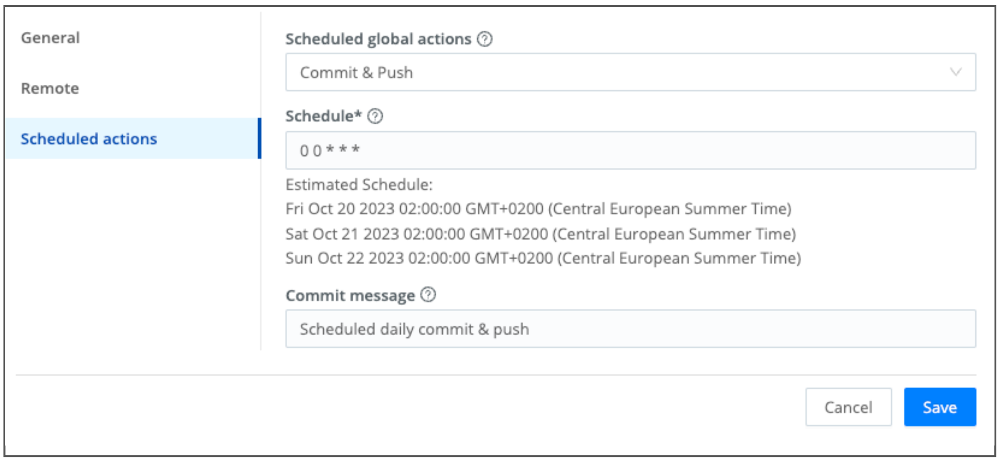
GitHub (specifically) does not support Basic authentication.

On the **Git Settings** > **Scheduled Actions** tab, you can schedule a **Commit**, **Push**, or **Commit & Push** action to occur on a predefined interval.



Git Scheduled Actions selection

For the selected action type, you can define a cron schedule, and a commit message distinct from the **General** tab's **Default Commit Message**. Then click **Save**.


General
Remote
Scheduled actions

Scheduled global actions: Commit & Push

Schedule*: 0 0 * * *

Estimated Schedule:
Fri Oct 20 2023 02:00:00 GMT+0200 (Central European Summer Time)
Sat Oct 21 2023 02:00:00 GMT+0200 (Central European Summer Time)
Sun Oct 22 2023 02:00:00 GMT+0200 (Central European Summer Time)

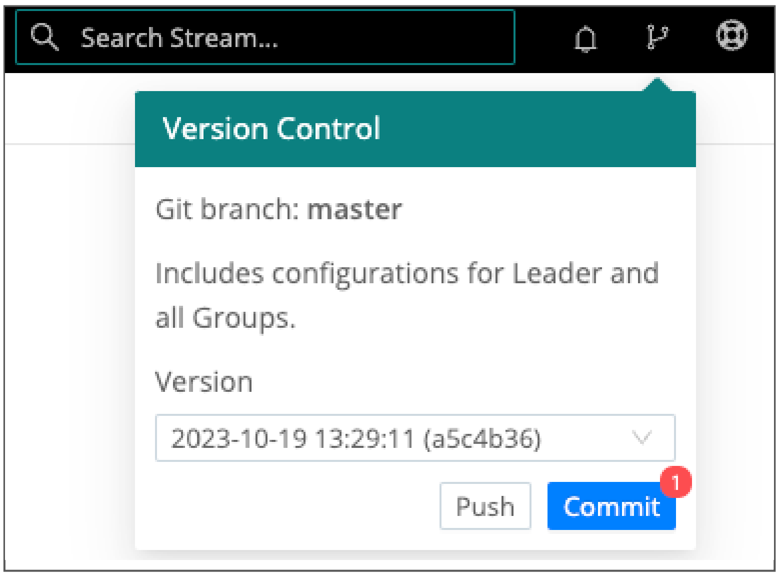
Commit message: Scheduled daily commit & push

Saving a Git Scheduled Action

You can schedule only one type of action. To swap to a different type, select it from the **Scheduled global actions** drop-down, and resave. To turn off scheduled Git commands, select **None** from the drop-down, and resave.

Pushing Configs to a Remote Repo

Once you've configured a remote, a **Git Push** button appears in the **Global Config** dialog. You can use this to copy committed configuration changes to your remote.



Version Control

Git branch: **master**

Includes configurations for Leader and all Groups.

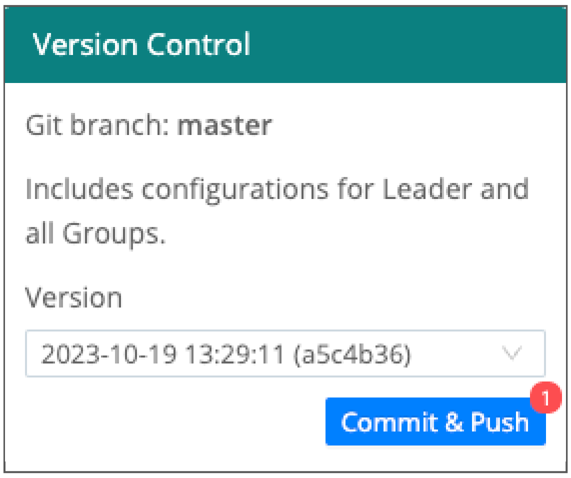
Version: 2023-10-19 13:29:11 (a5c4b36)

Git Push button

The branch indicator will normally read `master`, as shown above, unless you have enabled GitOps with an appropriate license.

Collapse Actions

If you enabled the **Git Settings** > **Collapse Actions** option, you will be offered a combined **Commit & Push** button in the **Global Config** dialog.


Version Control

Git branch: master

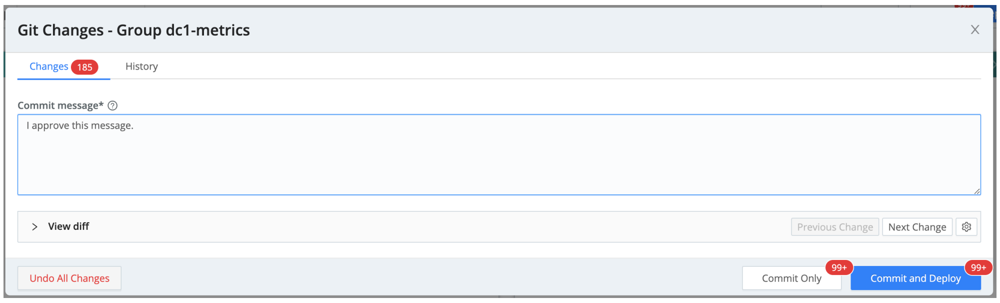
Includes configurations for Leader and all Groups.

Version

2023-10-19 13:29:11 (a5c4b36)

Git combined actions button

At the Group level, enabling **Collapse Actions** adds a combined **Commit & Deploy** option to the modal you see after clicking **Commit** for the Group's config.


Git Changes - Group dc1-metrics

Commit message*

I approve this message.

Git combined actions button for a Worker Group

Enabling **Collapse Actions** with a remote repo simplifies the **Commit Changes** confirmation dialog. It substitutes just a commit **Message** text box, with **OK** and **Cancel** buttons – omitting the diff view. Don't enable this option if you prefer to inspect configuration changes before committing.

Backing Up Configs Out of Band

Once you've configured your remote repo, changes to all Cribl Stream config files under the `$CRIBL_HOME` directory will normally be backed up to your remote whenever you click the above **Git Push** or **Commit & Push** UI buttons. The exceptions are:

- Any configuration files you've chosen to store outside `$CRIBL_HOME`.
- Any configuration files residing in paths you've added to `.gitignore`.

To back up these files to your remote, you'll need to either:

- Change the above paths/settings, or
- Use an external `git` client to manually push them.

Troubleshooting Push Errors

To resolve errors commonly encountered when pushing to a remote repo, see:

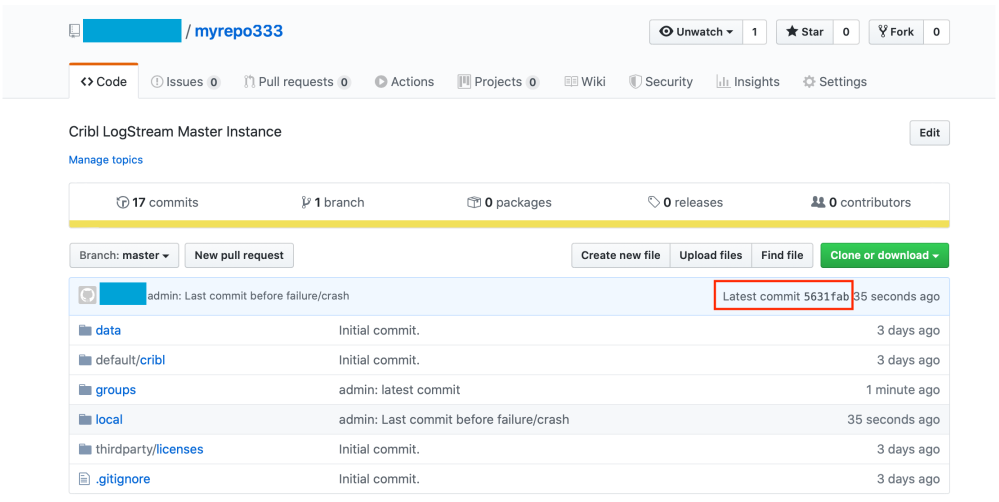
- Git Push Errors
- Git Remote Repos & Trusted CAs

Troubleshooting Resources

Cribl University offers a Troubleshooting Criblet on Commit & Deploy best practices and error recovery. 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.

Restoring Leader from a Remote Repo

If a remote repo is configured and has the latest known good Leader configuration, this section outlines the general steps to restore the config from that repo.



Cribl LogStream Master Instance

[redacted] admin: Last commit before failure/crash — Latest commit 5631fab 35 seconds ago

data	Initial commit.	3 days ago
default/cribl	Initial commit.	3 days ago
groups	admin: latest commit	1 minute ago
local	admin: Last commit before failure/crash	35 seconds ago
thirdparty/licenses	Initial commit.	3 days ago
.gitignore	Initial commit.	3 days ago

Restoring from remote repo

Let's assume that either the entire `$CRIBL_HOME` directory of the Leader is corrupted, or you're replicating the remote repo's config onto a fresh instance. Let's also assume that the remote repo has the form: `git@github.com:<username>/<reponame>.git`.

1. **Important**: In a directory of choice, untar the **same Cribl Stream version** that you're trying to restore, but do not start it.

Make sure you download the **matching** Cribl Stream version. Cribl's Download page foregrounds Cribl Stream's current version, but provides a link to this archive of prior releases.

2. Change directory into `$CRIBL_HOME` and initialize `git`:

```
# git init
```

3. If you are using SSH key authentication, specify the key using the following command:

```
GIT_SSH_COMMAND='ssh -i </path/to/github/repo>.key -o IdentitiesOnly=yes'
```

   As an alternative to executing `GIT_SSH_COMMAND` on the fly, you can set your key in `$CRIBL_HOME/.git/config` with:
   `git config core.sshCommand "ssh -i /path/to/key"`

Or set it globally with:
`git config --global core.sshCommand "ssh -i /path/to/key"`

4. Ensure that you have proper access to the remote repo:

```
# git ls-remote git@github.com:<username>/<reponame>.git
56331fabb4822eaec4ca0ffd008d6e9974c1e419f       HEAD
5631fabb4822eaec4ca0ffd008d6e9974c1e419f        refs/heads/master
```

5. Next, add/configure the remote:

```
# git remote add origin git@github.com:<username>/<reponame>.git
```

6. Now set up your local branch to exactly match the remote branch:

```
# git fetch origin
# git reset --hard origin/master
```

7. Finally, to confirm that the commits match, run this command while in `$CRIBL_HOME`:

```
# git show --abbrev-commit
```

You should see output indicating that `HEAD` points to both `master` and `origin/master`, as in this example:

```
commit 5631fab (HEAD -> master, origin/master)
Author: First Last <email@example.com>
Date:   Fri Jan 31 10:16:07 2020 -0500

    admin: Last commit before failure/crash
......
```

Step 6 above pulls in all the latest configs from the remote repo. Step 7 confirms the local repo matches the remote. You should now be able to start the Leader as normal. Once up and running, Workers should start checking in after about 60 seconds.

Verify cribl.secret

The `cribl.secret` file – located at `$CRIBL_HOME/local/cribl/auth/cribl.secret` – contains the secret key that is used to encrypt sensitive settings on configuration files (e.g., AWS Secret Access Key, etc.). Make sure this file is properly restored on the new Leader, because it is required to make encrypted conf file settings usable again.

.gitignore File

A `.gitignore` file specifies files that `git` should ignore when tracking changes. Each line specifies a pattern, which should match a file path to be ignored. Cribl Stream ships with a `.gitgnore` file containing a number of patterns/rules, under a section of the file labeled `CRIBL SECTION`.

.gitignore

```
# Do NOT REMOVE CRIBL and CUSTOM header lines!
# DO NOT REMOVE rules under the CRIBL section as they may be reintroduced on update
# You can ONLY comment out rules in the CRIBL section.
# You can add new rules in the CUSTOM section.
### CRIBL SECTION -- DO NOT REMOVE ###
default/ui/**
default/data/ui/**
bin/**
log/**
pid/**
data/uploads/**
diag/**
**/state/**
#### CUSTOM SECTION -- DO NOT REMOVE ###

<User-defined patterns/rules go here>
```

CRIBL Section

Do Not Remove CRIBL SECTION or CUSTOM SECTION Headers

Under the `CRIBL SECTION` header, Cribl Stream specifies the files that Git will ignore by default. Further files for Git to ignore should be listed under the `CUSTOM SECTION` header.

Do **not** add any lines to the `CRIBL SECTION` block, because the next update will just overwrite them anyway. Do not remove any lines either. The only modifications that will survive updates are commented lines.

Do not remove any section header lines, or unexpected behavior might occur on update.

CUSTOM Section

User-defined, custom patterns/rules can be **safely defined** under the `CUSTOM SECTION`. Cribl Stream will **not** modify the contents of `CUSTOM SECTION`.

Good candidates to add here include large lookup files – especially large binary database files. For details, see Git Push Errors: Large Files Detected.

Files skipped with .gitignore

If you have files that you've set `.gitgnore` to skip, you will need to back them up and restore them by means other than Git. For example, you can periodically copy/rsync them to a backup destination, and then restore them to their original locations after you complete the steps above.

Files specified in `.gitignore` are not only excluded from pushes to the remote repo, but are also excluded from Worker Group config bundles. When Workers load a new config that references a skipped (and missing) file, this can produce unexpected results, and usually errors.

For example, if you add `**/auth/certs/leader*` to `.gitignore`, this will prevent the `leader.crt` and `leader.key` files from leaking to Worker Nodes in config deployments.

Configs that you `.gitignore` in this way on the Leader will not persist on Workers after they receive their next deploy. However, these configs will be available in the next few rolling backup bundles in `state/`.

Commit and Deploy via the API

You can automate commit and deploy commands by using Cribl Stream API requests. The following examples show how to do so for a Worker Group's configuration.

About the Examples

- The API calls below include Worker Group names as path parameters.
- The `curl` commands assume that you have set the `$token` environment variable to match the value of a bearer token. See Authentication for alternative approaches; adapt the example commands to suit your chosen approach.

Commit Changes

To commit pending changes to your configured repo, adapt and run the following API call:

```
POST /api/v1/version/commit

{"message":"a descriptive commit message","group":"<worker_group_name>"}
```

You will receive a JSON response with some details about the commit:

```
{"items":[{"branch":"master","commit":"abcd1234","summary":{"changes":"1","inserti
```

From that response, you'll need to extract the commit ID (`abcd1234`) to use in the second deploy API call below.

Commit Example

Here, let's commit all pending changes to the default Worker Group:

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json"
```

Deploy Changes

To deploy your committed config changes to Worker Groups, adapt and run the following API call. As the `version` value, you'll need the commit ID you received in the commit response above:

```
PATCH /api/v1/master/groups/<worker_group_name>/deploy

{"version":"abcd1234"}
```

A successful response payload looks like this:

```
{"items":[{"description":"Default Worker Group","tags":"default","configVersion":"
```

Deploy Example

Here, let's deploy the previously committed configuration to the `default` Worker Group:

```
curl -X PATCH -H "Authorization: Bearer $token" -H "content-type: application/json
```

Selectively Commit Changes

If you want to commit only a subset of configuration changes, adapt and use the following payload:

```
POST /api/v1/version/commit

{
  "message":"descriptive commit message",
  "group":"default",
  "files":["groups/default/data/samples/sample-g0aT.json","groups/default/local/cr
}
```

Selective Commit Example

Let’s selectively commit a sample data file, and the updated YAML listing of all sample files, to the `default` Worker Group:

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json"
```

9.3. GitOps

With Cribl Stream's GitOps features, you can manage Cribl Stream configuration with standard version-control systems and CI/CD flow. You can separate development from production configurations, and thus, safely build and continuously deploy your observability pipelines. The production environment will be read-only, and thus strongly protected against unauthorized or unintended changes. This intentionally restrictive approach is a good fit for some but not all organizations.

> The GitOps workflow described here assumes that your development, staging, and production environments are identical or nearly identical. If, on the other hand, your development environment differs significantly from production (in its branching structure, for example, or if users have wide latitude for experimentation), you should consider solutions other than GitOps. If you have questions about how GitOps compares to various alternatives, please contact Cribl Support.
>
> GitOps requires an Enterprise license and a distributed Cribl Stream deployment.

> Cribl.Cloud does not support GitOps.

How to Use This Doc

This doc has three sections.

- Work through all three sections, in order, once.
- Once you have done that, you are finished with the first two sections, Preparing Your Git Repo and Preparing Cribl Stream.
- Next, incorporate the HTTP request from the third section (GitOps Workflow) into your CI/CD tool infrastructure.
- From then on, your normal production routine will be to just repeat the GitOps Workflow (that is, the third section) again and again.

You'll perform setup steps in the Cribl Stream UI, in your Git management system, and at the command line. Screenshots are in light mode for Cribl Stream, and in dark mode for GitHub examples.

Environment and branch names have been chosen to help you keep track of what you're doing – if you decide to use different ones, exercise care.

Procedures begin with instructions that are agnostic to the Git management system you want to use, whether that is free GitHub, GitHub Enterprise, GitLab, Bitbucket, or a self-hosted Git server. The detailed instructions that follow are for use with free GitHub. If you're using one of the other Git management systems, you'll need to substitute appropriate instructions for the ones that begin "To do this in GitHub ..."

Check out Cribl's free GitOps sandbox for a "deep dive" and to work through a practical example.

If you encounter either of the following situations, see the linked appendix:

- You need to upgrade your GitOps-enabled Cribl Stream deployment.
- You are not using systemd, and need to override the defaults using environment variables.

Preparing Your Git Repo

The procedure in this section only needs to be done once. When you've completed it, you can prepare Cribl Stream for GitOps.

Creating SSH Public and Private Keys

You'll use SSH to connect to your Git management system; Cribl does **not** recommend using basic authentication for GitOps.

At the command line, generate public and private SSH keys, specifying Ed25519 as the key type:

```
ssh-keygen -t ed25519 -C "your_email@example.com"
```

Verify that the `~/.ssh/` directory contains both of the newly-generated keys:

- `id_ed25519` (the private key that Cribl Stream will use).
- `id_ed25519.pub` (the public key that your Git management system will use).

Copy the public SSH key to your clipboard:

```
cat ~/.ssh/id_ed25519.pub
```

Associate the public key with your Git management account.

To do this in GitHub:

1. If you do not have a GitHub account, create one.
2. Sign in to your GitHub account.
3. On the SSH and GPG keys page, click **New SSH key**.

4. Paste the key in the **Key** field.
5. Click **Add SSH key**.

Creating a Private Git Repo

You need to create a dedicated, private Git repo – your Development environment will write to it, and your Production environment will sync to it. Create a private repo named `cribl` that contains one branch, named `main`, now.

To do this in GitHub:

1. In your GitHub profile page, click **Go to your personal profile**.
2. On the resulting page, select the **Repositories** tab.
3. Click **New** to create a new repository.
4. Name the repo `cribl`.
5. Change the repo from **Public** to **Private**.
6. Click **Create repository**. The new repo's landing page will open.
7. In the Quick setup box, click **creating a new file** to open the text editor.

   > ⚠ This step and the next are **required** because they establish the commit history that procedures in the next section depend on.

8. Enter some text, enter a name for the file, and click **Commit changes** (once in the text editor and again in the pop-up).

By default, the repo will contain one branch named `main`.

Setting Up the `prod` Branch

We've chosen names that always make it clear whether we're talking about environments or about branches:

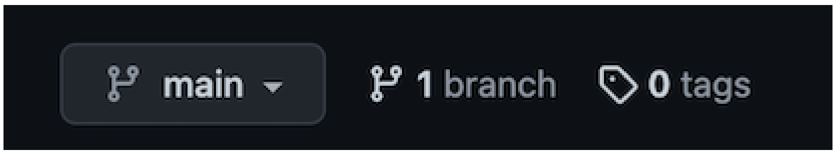
- Development and Production will be your Cribl Stream environments.
- `dev` and `prod` will be the branches in your Git repo.

In this section, we'll set up `prod`; later on, we'll set up `dev`.

Recall that your Git repo has one branch, named `main`. Now, rename `main` as `prod`.

To do this in GitHub:

1. In your GitHub repo's landing page (e.g., `https://github.com/<your_Git_username>/cribl`), select the **1 branch** tab. (That's what the branches tab will be named, assuming that the sole branch is `main`).



The branches tab

2. In the `main` branch's row, click the pencil icon at far right, and rename `main` as `prod`.

Preparing Cribl Stream

The procedure in this section only needs to be done once. When you've completed it, you can move on to testing and using GitOps.

Setting Up Your Environments

In your distributed Cribl Stream deployment, set up two separate environments, one named `Development` and the other named `Production`. Cribl strongly recommends that you use systemd to manage the `cribl` service.

Complete the following steps in **both** environments:

- Create one Cribl Leader and two Cribl Workers.
- On the Leader, install a Cribl Enterprise License.
- On the Leader, install `git` using the following command (or the equivalent for your OS):
  `sudo yum install git -y`

Then, in the Production environment:

- On the Leader, install `jq` – the GitOps workflow instructions below use `jq`.
  - Run this command (or the equivalent for your OS):
    `sudo yum install jq -y`
- Note the Production Leader's IP address for use later on.

Configuring Your Production Leader

First, you'll ensure that your Production Leader and your Git repo can communicate; then, you'll put your Production Leader into read-only mode.

Creating the `gitops` User

1. Create a new local user called `gitops`.
2. On the Production Leader's local filesystem, temporarily record the password for the `gitops` user; we'll refer to it as `GITOPS_USER_PASSWORD`.
3. Associate the default `gitops` Role with the `gitops` user.
    - This Role's only permission is to `POST` to the `/version/sync` endpoint of the Cribl API (in the API Reference, see **versioning**).

Adding a Secret to Git

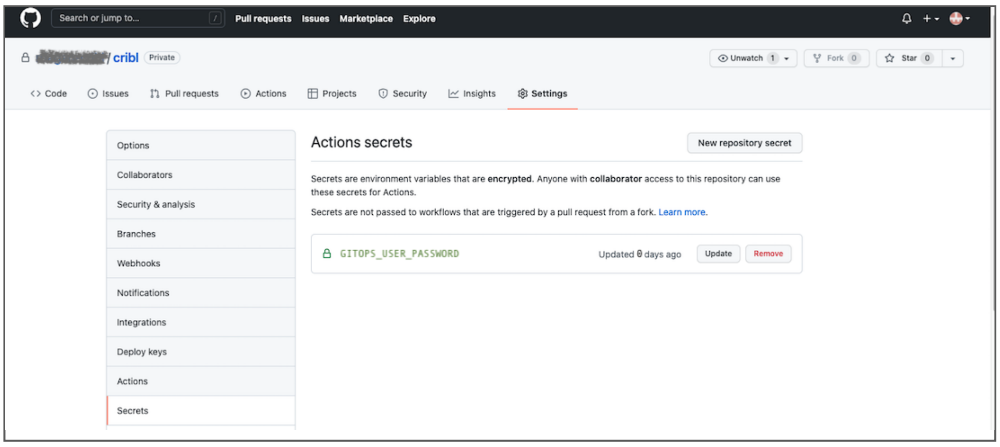
Your Git repo needs to be able to communicate with your Production Leader. This requires that you store a secret in your Git management system. We'll use `GITOPS_USER_PASSWORD` as the secret.

From the Production Leader, upload the `GITOPS_USER_PASSWORD` to the Git management system (or perform an equivalent operation) now.

To do this in GitHub:

1. In the GitHub site for your repo, navigate to **Settings** > **Secrets and variables**.
2. Click **New repository secret** and follow the prompts.

When done, you should see a `GITOPS_USER_PASSWORD` row under **Actions secrets**:



Actions secrets

Secrets are environment variables that are **encrypted**. Anyone with **collaborator** access to this repository can use these secrets for Actions.

Secrets are not passed to workflows that are triggered by a pull request from a fork. Learn more.

`GITOPS_USER_PASSWORD` Updated 0 days ago

The GitOps user password has been uploaded to GitHub

Connecting Your Production Environment to Your Git Repo

In Cribl Stream, in the UI for the Production Leader, select global **Settings** (lower left) > **Git Settings**.

In the **Remote** tab:

1. In **Remote URL**, enter the URL for your private Git repo.
   - On GitHub, the **Remote URL** will look like this: `git@github.com:<your_GitHub_username>/cribl.git`
2. Make sure that in the **Authentication type** drop-down, `SSH` (the default) is selected.
3. In **SSH private key**, copy and paste the private key (`~/.ssh/id_ed25519`).

   The key will paste in with an appended newline below it. Do not delete this newline before you save the Remote Settings, or else you will trigger an error of the form: `fatal: Could not read from remote repository.`

4. Toggle **SSH strict host key checking** off. This will allow the remote repository to accept your host key information. If you miss this step, you will trigger an error.
5. Click **Save**.
6. Toggle **SSH strict host key checking** on again.
7. Click **Save** (again!).
8. **Commit** and **Push**.

In the **General** tab:

1. Set **Branch** to `prod`.
2. Leave **GitOps Workflow** set to `None`.
3. Click **Save**.

Check your private Git repo – in the `prod` branch, you should now see your commit.

Making the Production Leader Read-Only

In a terminal on the Production Leader, become root:

```
sudo su -
```

Add `systemctl` overrides to set environment variables that will (a) make the Production Leader read-only upon startup, and (b) set the branch to `prod`.

1. Use `systemctl` to open the `override.conf` file in an editor:

```
systemctl edit cribl
```

2. When prompted, add the following overrides:

```
[Service]
Environment=CRIBL_GIT_OPS=push
Environment=CRIBL_GIT_BRANCH=prod
```

3. Restart:

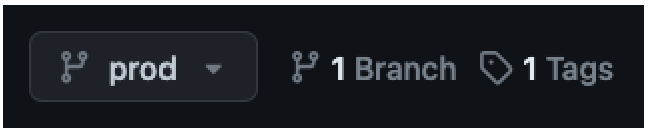
```
systemctl daemon-reload && systemctl restart cribl
```

Creating the dev Branch

Recall that your Git repo has one branch, named `prod`. Now, create a new branch named `dev`, **from your existing** `prod` branch.

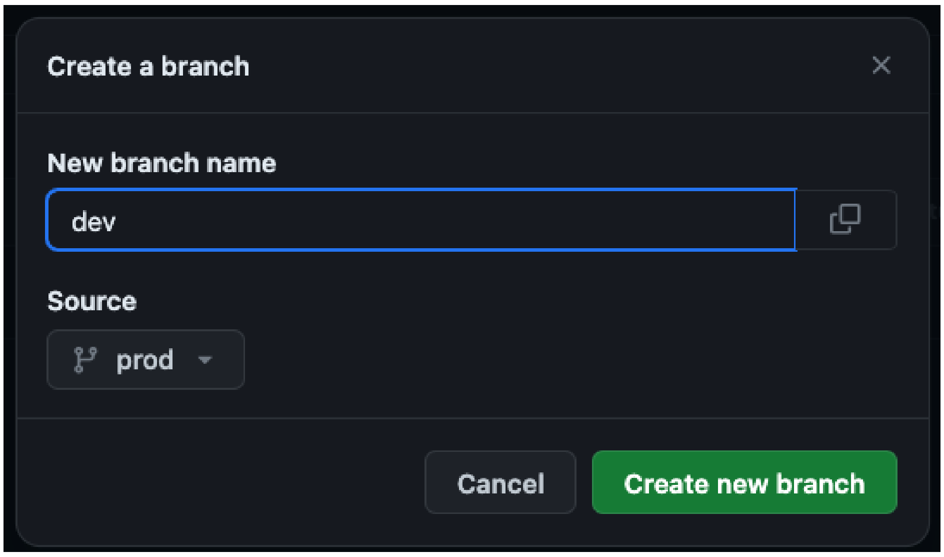
To do this in GitHub:

1. In your GitHub repo's landing page (e.g., `https://github.com/<your_Git_username>/cribl`), select the **1 branch** tab. (That's what the branches tab will be named, assuming that the sole branch is `prod`).



The branches tab

2. Click **New branch** to open the **Create a branch** modal.
3. Make sure that **Source** is set to `prod`.
4. In the **New branch name** field, enter `dev`.



Creating the `dev` branch

5. Click **Create new branch** to close the modal.
6. Back in the **Branches** tab, click **Overview** to refresh the tab – you should now see both the `prod` and `dev` branches listed.

Configuring Your Development Leader

First, you'll point your Development Leader to the `dev` branch in the remote repo. Then, you'll ensure that one or more users exist that can push configs to that repo.

Pointing Your Development Leader to Its Branch

In Cribl Stream, in your Development Leader's UI, select global **Settings** (lower left) > **Git Settings**.

In the **Remote** tab:

1. In **Remote URL**, enter the URL for your private Git repo – and if the URL includes a Git username, this should be `gitops`, the one you created above.
   - On GitHub, the **Remote URL** will look like this: `git@github.com:<your_GitHub_username>/cribl.git`
2. Make sure that in the **Authentication type** drop-down, `SSH` (the default) is selected.
3. In **SSH private key**, copy and paste the private key (`~/.ssh/id_ed25519`).

> The key will paste in with an appended newline below it. Do not delete this newline before you save the Remote Settings, or else you will trigger an error of the form: `fatal: Could not read from remote repository.`

4. Toggle **SSH strict host key checking** off. This will allow the remote repository to accept your host key information. The remote host's information is stored in `~/.ssh/known_hosts`. If you miss this step, you will trigger an error.
5. Click **Save**.
6. Toggle **SSH strict host key checking** on again.
7. Click **Save** (again!), but do not **Commit** or **Push** yet.

In the **General** tab:

1. Set **Branch** to `dev`.
2. Leave **GitOps Workflow** set to `None`.
3. Click **Save**, but do not **Commit** or **Push** yet.

Restart the Development Leader. When it comes back up, it will be able to write to your Git repo.

Setting Up Users to Push to the Remote Repo

You will need one or more users capable of pushing configs to the remote repo. There is more than one way to achieve this. Your choice will depend on the needs of your organization.

- Any user with the `admin` role already has ability to push to the remote repo. If this is your preferred approach, skip to the next section.
- If you prefer to allow one or more **regular** users to push to the remote repo, **without** having the power of the `admin` role, you can create such users by completing the procedure below.

First, edit the `roles.yml` file to make a new `push_to_remote` role. (We're using the command line here because you cannot perform this operation in the UI.)

1. SSH into the Leader node.
2. `cd` into your `$CRIBL_HOME/local/cribl` directory.
3. Open or create the `roles.yml` file in a text editor.
4. Without changing the existing text in the file, paste in the following snippet:

```
    push_to_remote:
      policy:
        - POST /version/sync
        - POST /version/commit
        - POST /version/push
        - GET /system/settings/git-settings
      description: Members with this role can push configs to a remote repo.
```

5. Restart Cribl Stream.

Next, back in the UI, create a new `config_pusher` user.

1. In global **Settings** (lower left) > **Access Management** > **Local Users**, click **Add User** to open the user creation modal.
2. Name the new local user `config_pusher`.
3. In addition to the usual configuration steps, use the **Roles** drop-down to associate the `push_to_remote` role with the `config_pusher` user.
4. Click **Save**.
5. **Commit** and **Push**.

Verify That Your New User Can Push to the Remote Repo

1. Log out, then log back in as `config_pusher`.
2. The **Version Control** drop-down should now display a **Push** button.

Later, when you are ready to push new configs to the remote repo, click this new **Push** button.

GitOps Workflow

If you've completed the above preparatory sections, your GitOps workflow is ready go. Here's an overview, with links to detailed procedures.

- Work in the Development Environment: All your commits and testing happen in the Development environment first; you push changes to the `dev` branch in your Git repo.
- Merge Changes from dev to prod: When you are satisfied that your changes are ready for production, you create a pull request that brings the `prod` branch up-to-date with `dev`. This is all in your Git management system, and changes nothing in your Cribl Stream Production environment.
- Sync Your Production Leader to Git: For your changes to take effect in production, you will send an HTTP request to the `/version/sync` endpoint of the Cribl API, on the Production Leader. Once you

are in production, it will be your CI/CD infrastructure – not you – that sends the request. This will cause the Production Leader to pull in the latest changes from your Git repo's `prod` branch – without violating the Production Leader's read-only status.

The procedures include a harmless test task so that you can validate your setup and acquire a feel for the workflow.

Working in the Development Environment

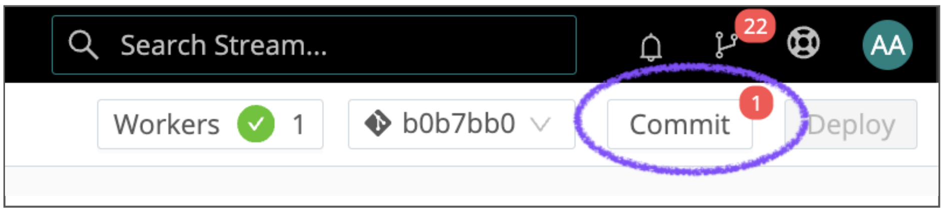
In this step, you'll make changes in the Development environment, and subsequently propagate them to Production. To test this with a minimal unit of work, start by adding a Route.

In Cribl Stream, in the Development Leader's UI, navigate to **Routing** > **Data Routes**.

1. Click **Add Route**.
2. Configure the new Route as follows:

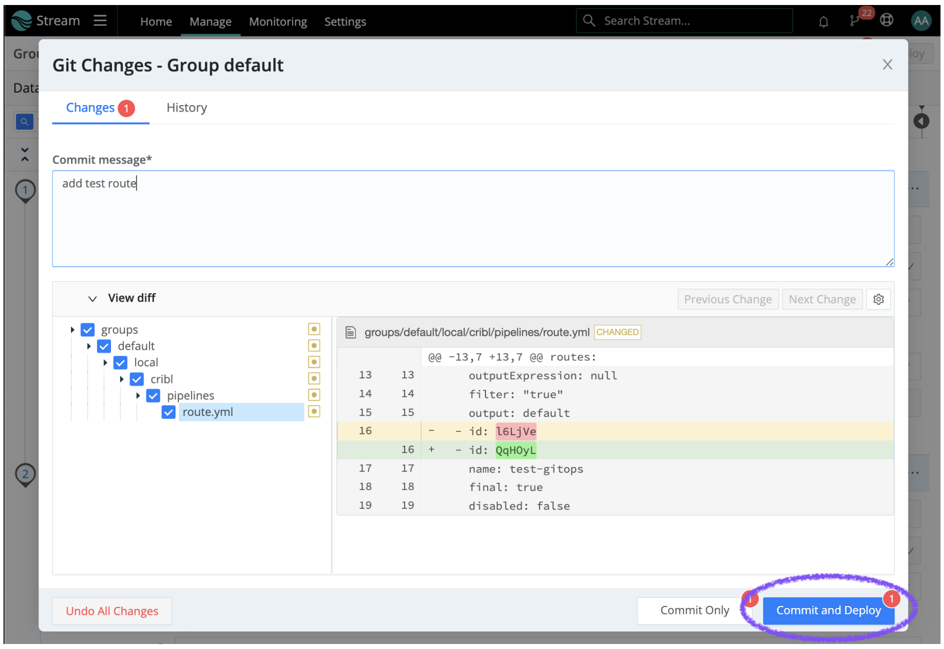
Name	Filter	Pipeline	Output
`test-gitops`	`true`	`devnull`	`devnull:devnull`

3. Click **Commit** as shown in the screenshot below.



Adding the new Route: initial Commit

4. When the **Git Changes** modal opens, add a commit message and click **Commit and Deploy** as shown in the screenshot below.



```
groups/default/local/cribl/pipelines/route.yml CHANGED
@@ -13,7 +13,7 @@ routes:
      outputExpression: null
      filter: "true"
      output: default
-   - id: l6LjVe
+   - id: QqHOyL
      name: test-gitops
      final: true
      disabled: false
```

Adding the new Route: Commit message, Commit, and Deploy

5. Finally, push the changes to your remote repo via the **Global Config** dialog, whose appearance will vary depending on how you've configured **Git Settings** > **Collapse Actions**. Make sure the dialog says that the **Git branch** you're pushing to is `dev`, as configured above.

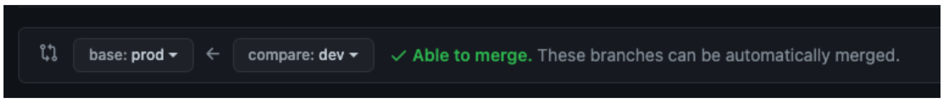
Merging Changes from `dev` to `prod`

In your Git management system's `dev` branch, you should see the new `route.yml` appearing in the `/groups/<group_name>/local/cribl/` directory.

Merge the changes from the `dev` branch to `prod` now.

To do this in GitHub:

1. Click **Compare & pull request**. Be sure that the `base` drop-down says `prod` and the `compare` drop-down says `dev`.


base: prod
compare: dev
✓ Able to merge. These branches can be automatically merged.

Comparing `dev` to `prod`

2. Add some comments.
3. Click **Create pull request**. The UI should say that you want to `merge 1 commit into prod from dev`.
4. If GitHub detects conflicts, fix them.
5. Have the appropriate person on your team review the pull request.

6. You (or the appropriate person) should now click **Merge pull request**, then **Confirm merge**.

If the merge is successful, GitHub will display a confirmation message. Your committed changes have now been merged from `dev` to `prod`.

Syncing Your Production Leader to Git

Your changes so far have all been done in the Development environment. Meanwhile, you have committed to the `dev` branch, and pushed to the `prod` branch **in your Git management system** – but your Production **environment** has not changed.

Since, for safety's sake, you made the Production environment read-only, you can neither write to it nor update it via a UI. Instead, the following must happen:

- You will send an HTTP request to the `/version/sync` endpoint of the Cribl API, on the Production Leader (in the API Reference, see **versioning**). Once you are in production, it will be your CI/CD infrastructure – not you – that sends the request.
- This will cause the Production Leader to pull in the latest changes from your Git repo's `prod` branch – without violating the Production Leader's read-only status.

Below you'll see a script you can use to send the HTTP request.

- This script is generic, and will require you to adapt it to work with the Git management system you're using.
- Once you run the script, you should see the changes you made in your Development environment (in this example, the new Route) in the Production environment, too.

Syncing to Git: Generic Script

Substitute your Production Leader's IP address and the `gitops` user's password for the placeholders.

```
#!/bin/bash

# Authenticate and Save the Token.
TOKEN=$(curl http://<Production_Leader_IP_address>:9000/api/v1/auth/login \
     -H 'Content-Type: application/json' \
     -d '{"username":"gitops","password":"$GITOPS_USER_PASSWORD"}' 2>/dev/null | \
     jq -r .token)

# Set up the Authentication Header
export AUTH_HEAD="Authorization: Bearer $TOKEN"

# Make the notification call
curl -X POST "http://<Production_Leader_IP_address>:9000/api/v1/version/sync" -H "a
```

Making the HTTP Request from CI/CD

Now that you have run one of the above scripts to prove that your GitOps workflow works, you should incorporate the script's HTTP request into your CI/CD setup. Once that is done, you're in production with GitOps! To develop, test, and push changes to production, just keep running through the whole GitOps workflow.

Appendix: Upgrading

Upgrading a Cribl Stream GitOps deployment is essentially a variation on the day-to-day GitOps workflow described in the previous section, with the key difference that you take the Production environment out of, then put it back into, read-only mode.

Once you have begun the upgrade, **do not** make any changes unrelated to the upgrade process, in either the Development or Production environment, until the upgrade process is complete. For example, do not start creating any new Routes or Pipelines.

First, in your Development environment:

1. Validate that Cribl Stream is working as expected, and that the latest changes have been pushed to the `dev` branch in your Git repo.
2. Stop the Cribl Stream Leader and back up `$CRIBL_HOME` (the Cribl install directory).
3. Use the UI to upgrade the Leader and Workers.
4. Create a PR to merge the `dev` branch into `prod`. Merge the PR.

Next, in your Production environment:

1. Stop the Cribl Stream Leader and back up `$CRIBL_HOME` (the Cribl install directory).
2. Take your Production environment **out of read-only mode** according to the instructions below.
3. Use the UI to upgrade the Leader and Workers – but **do not** commit changes after the upgrade.
4. Put the Leader **back into read-only mode** according to the instructions above.
5. Sync the Production environment to the `prod` branch in your Git repo.

Finally, confirm that all Workers in both environments have the latest configuration.

Taking the Production Leader Out of Read-Only Mode

In a terminal on the Production Leader, become root:

```
sudo su -
```

Comment out the `systemctl` overrides that set environment variables that (a) had made the Production Leader read-only upon startup, and (b) had set the branch to `prod`.

1. Use `systemctl` to open the `override.conf` file in an editor:

   ```
   systemctl edit cribl
   ```

2. When prompted, comment out the Git-related overrides as shown:

   ```
   [Service]
   # Environment=CRIBL_GIT_OPS=push
   # Environment=CRIBL_GIT_BRANCH=prod
   ```

3. Restart:

   ```
   systemctl daemon-reload && systemctl restart cribl
   ```

Appendix: Environment Variables

If you are not using systemd to manage the `cribl` service, you can use environment variables to override the defaults in the UI.

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_DEPLOY	Controls whether to deploy Workers or not.
CRIBL_GIT_AUTH	One of: `none`, `basic`, or `ssh`.
CRIBL_GIT_USER	Used for `basic` auth.
CRIBL_GIT_PASSWORD	Used for `basic` auth.
CRIBL_GIT_OPS	Controls which GitOps workflow to use – either `push` or `none`.

9.4. Worker Group Settings

A Worker Group's General Settings allows you to configure teleporting, throughput throttling, logging, upgrading, and security at the Worker Group level.

To edit a Worker Group's General Settings:

1. Select the Worker Group, then click **Worker Group Settings**.
2. Configure the following settings per Worker Group.

General Settings

Worker Group Configuration

You can configure the following options on this tab:

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

Tags: Use tags to organize Worker Groups into logical categories. Then, you can search by tags on the **View all Groups** (Stream) or **View all Fleets** (Edge) interface. This search filters the list of Worker Groups, showing only those with the tag you entered.

Enable teleporting to Workers: Use this toggle to enable or disable authenticated access to Workers' UI from the Leader (Stream, Edge).

In Cribl Stream 4.1.2 and later, you can configure the following option in **Shutdown Settings**.

API Server Settings

General

You can set the following options for the API server. In **General Settings**, open **API Server Settings** and select **General**.

Host: The hostname or IP address you want to bind the API server to. Defaults to `0.0.0.0`.

Port: API port to listen to. Defaults to `9000`.

TLS

For information on TLS options, see the documentation for any Source or Destination that supports TLS.

Advanced

See Authentication Controls.

Default TLS Settings

See the Securing and Monitoring topic.

Limits

The **Limits** tab provides access to controls for metrics, storage, metadata, jobs, the Redis cache and connections to it, and CPU settings.

Metrics

See Controlling Metrics Volume.

Storage

You can configure the following options for storage. In **General Settings**, open **Limits** and then select **Storage**:

Max sample size: Maximum file size, in binary units (`KB`, `MB`), for sample data files. Maximum: `3 MB`. Default: `256 KB`.

Min free disk space: The minimum amount of disk space on the host before various features take measures to prevent disk usage (KB, MB, etc.). Default: `5 GB`.

Max PQ size per Worker Process: Highest accepted value for the **Max queue size** option used in individual Sources' and Destinations' persistent queues. Default: `1 TB`. Consult Cribl Support before increasing beyond this value.

Metadata

Event metadata sources: List of event metadata sources to enable. No sources are enabled by default.

Jobs

Disable jobs/tasks: Set to `Yes` by default in Edge Fleets and `No` by default in Stream Groups. In Edge, Nodes no longer poll the Leader for upgrade jobs. Setting this to `Yes` in Edge reduces application load from the Leader.

> Nodes running 4.4.4 and older still upgrade via jobs and will honor the Jobs settings, even with the **Disable jobs/tasks** toggle enabled.

Job Limits

Disable Jobs/Tasks: When enabled, the Worker Nodes won't poll the Leader for jobs/tasks. The job limits settings below will not affect Worker Nodes on version 4.5.0 and newer. Worker Nodes running 4.4.4 and older still use these jobs settings even if jobs/tasks are disabled here.

Concurrent Job Limit: The total number of jobs that can run concurrently. Defaults to `10`.

Concurrent System Job Limit: The total number of **system** jobs that can run concurrently. Defaults to `10`. Minimum `1`.

Concurrent Scheduled Job Limit: The total number of **scheduled** jobs that can run concurrently. This limit is set as an offset relative to the **Concurrent Job Limit**. Defaults to `-2`.

> Skipped jobs indicate that a Group's **Concurrent Job Limit** has been reached or exceeded. Increase this limit to reduce the number of skippable jobs. For resource-intensive jobs, this might require deploying more Worker Nodes.

Task Limits

Concurrent Task Limit: The total number of tasks that a Worker Process can run concurrently. Defaults to `2`. Minimum `1`.

Concurrent System Task Limit: The number of system tasks that a Worker Process can run concurrently. Defaults to `1`. Minimum `1`.

Max Task Usage Percentage: Value, between `0` and `1`, representing the percentage of total tasks on a Worker Process that any single job may consume. Defaults to `0.5` (i.e., 50%).

Task Poll Timeout: The number of milliseconds that a Worker's task handler will wait to receive a task, before retrying a request for a task. Defaults to `60000` (i.e., 60 seconds). Minimum `10000` (10 seconds).x

Completion Limits

Artifact Reaper Period: Interval on which Cribl Stream attempts to reap jobs' stale disk artifacts. Defaults to `30m`.

Finished Job Artifacts Limit: Maximum number of finished job artifacts to keep on disk. Defaults to `100`. Minimum `0`.

Finished Task Artifacts Limit: Maximum number of finished task artifacts to keep on disk, per job, on each Worker Node. Defaults to `500`. Minimum `0`.

Task Manifest and Buffering Limits

Manifest Flush Period: The rate (in milliseconds) at which a job's task manifest should be refreshed. Defaults to `100` ms. Minimum `100`, maximum `10000`.

Manifest Max Buffer Size: The maximum number of tasks that the task manifest can hold in memory before flushing to disk. Defaults to `1000`. Minimum `100`, maximum `10000`.

Manifest Reader Buffer Size: The number of bytes that the task manifest reader should pull from disk. Defaults to `4kb`.

Job Dispatching: The method by which tasks are assigned to Worker Processes. Defaults to `Least In-Flight Tasks`, to optimize available capacity. `Round Robin` is also available.

Job Timeout: Maximum time a job is allowed to run. Defaults to `0`, for unlimited time. Units are seconds if not specified. Sample entries: `30`, `45s`, `15m`.

Task Heartbeat Period: The heartbeat period (in seconds) for tasks to report back to the Leader/API. Defaults to `60` seconds. Minimum `60`.

Redis

Cache

Key TTL in seconds: Maximum time to live of a key in the cache (seconds). `0` indicates no limit. Defaults to `10 minutes`.

Max # of keys: Maximum number of keys to retain in the cache. `0` indicates no limit. Defaults to `0`.

Max cache size (bytes): Maximum number of bytes to retain in the cache. `0` indicates no limit. Defaults to `0`.

Service period (seconds): Frequency of cache limit enforcement. Defaults to every `30 seconds`.

Server assisted: Defaults to `No`. When toggled to `Yes`, the following control appears.

Client tracking mechanism: Mechanism for invalidation message delivery. In default mode, the server remembers which keys a client has requested and only sends invalidations for those, using more Redis server memory. In broadcast mode, it sends all invalidations, requiring more processing by Cribl Stream.

Connections

Reuse Redis connections: Toggle on if you want Cribl Stream to try to reuse Redis connections when multiple Redis Functions (or references to them) are present. When enabled, displays the following additional control:

- **Max number of connections**: The maximum number of identical connections allowed before Cribl Stream tries to reuse connections. Defaults to `0`, meaning unlimited connections are allowed (equivalent to leaving **Reuse Redis connections** toggled off). Setting a non-zero integer value forces Cribl Stream to try to reuse connections for each individual Worker Process (**not** to reuse connections **among** Worker Processes).

To understand why and when to employ these controls, see Reusing Redis Connections.

Other

CPU profile TTL: The time-to-live for collected CPU profiles.

Default managed node heartbeat period: How many seconds a managed Node will wait to send back a heartbeat to the Cribl control plane.

Proxy Settings

Use proxy env vars: Honors the `HTTP_PROXY/HTTPS_PROXY` environment variables. Defaults to `Yes`.

Sockets

Directory: Holds sockets for inter-process communication (IPC), such as communications between a load-balancing process and a Worker Process. Defaults to `/tmp` (your system's temp directory).

Shutdown Settings

Drain timeout (sec): Determines how long a Cribl server will wait for writes to complete before the server shuts down on individual Worker Processes. If you notice that Workers are under-ingesting available data upon shutdown or restart, increase the `10`-second default. Acceptable range of values: minimum `1` second, maximum `600` seconds (10 minutes).

Worker Processes

For details about this left tab's **Process count**, **Minimum process count**, and **Memory (MB)** controls, see Sizing and Scaling.

Process count and **Minimum process count** are not configurable on Cribl Edge, where each Edge Node is automatically allocated `1` Worker Process.

The following controls are also available on this tab to optimize Worker Processes' throughput on startup.

Max connections at startup: Maximum number of connections accepted at Worker Process startup. Defaults to `1`. Enter a negative integer for unlimited connections.

Startup throttling duration (ms): Maximum time (in milliseconds) to continue throttling connections after Worker Process startup. Defaults to `10000` ms (10 sec.) Enter `0` to disable throttling.

Load throttle %: Sets a threshold to prevent overwhelming Workers. If 90% of a Worker Process' CPU utilization readings exceed this threshold over one minute, the process will reject new connections until the CPU load stabilizes. Another process that is below the threshold will accept the connection the next time it is established. Defaults to `0`% (no throttling). Enter a percentage between `1-100` to enable throttling.

You can configure the CPU saturation threshold, but the 90% sampling trigger is not configurable. Also, `_raw stats` > `cpuPerc` values might diverge from your **Load throttle %** threshold. This is because `cpuPerc` is sampled and averaged once per minute, whereas the **Load throttle %** is evaluated every second, with a rolling 1-minute lookback sample. (These intervals are also not configurable.)

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.

9.5. Notifications Overview

Notifications alert Cribl Stream admins about issues that require their immediate attention.

This page describes the uses for Cribl Stream Notifications, their license requirements, and how to configure them.

Types of 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. For details, see Source-State Notifications.
- Sources and Collectors that report no data flow. For details, see Source-State Notifications.
- Destinations experiencing backpressure. For details, see Destination-State Notifications.
- Destinations approaching their persistent queue threshold. For details, see Destination-State Notifications.
- Destinations that report errors. For details, see Destination-State Notifications.
- Pending expiration of a Cribl Stream license. For details, see License-Expiration Notifications in Cribl.

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

License Requirements

Notifications require an Enterprise or Standard license. Without an appropriate license, the configuration options described below will be hidden or disabled in Cribl Stream's UI.

Notifications and Targets

A Notification target specifies the delivery method for a Notification. Every Notification requires one or more targets.

Available target types include:

- Webhook

- PagerDuty integration
- Slack
- AWS SNS
- Email

For details, see Notification Targets.

By default, any Notification that you configure will have a target of `System Messages`. When a Notification condition is triggered, Cribl Stream will add an indicator on the top nav's **Messages** button. Click this button to view details in the **Messages** drawer.

Notifications and RBAC

Notifications work with Cribl Stream'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, configure Notifications, and configure targets.

Implementing Notifications

Implementing a Notification is a three-step process:

- Create a Notification about a Source state, Destination state, or pending license expiration.
- Add or create a Notification target.
- Test the target to be sure it will work when needed.

You can create either the Notification or the target first — the order doesn't matter.

To create a Notification about a Source or Destination state, click **Add Notification** in the Source or Destination modal, **Notifications** tab.

To create a license-expiration Notification, click **Add Expiration Notification** in the **Global Settings** > **Licensing** UI.

General

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

Configuration

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

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

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

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

Metadata

Metadata fields are user-defined fields included in the notification payload. All Notification types can contain metadata.

Click **Add field** here to add custom metadata fields to your Notifications in the form of key-value pairs:

- **Name**: Enter a name for this custom field.
- **Value**: Enter a JavaScript expression that defines this field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

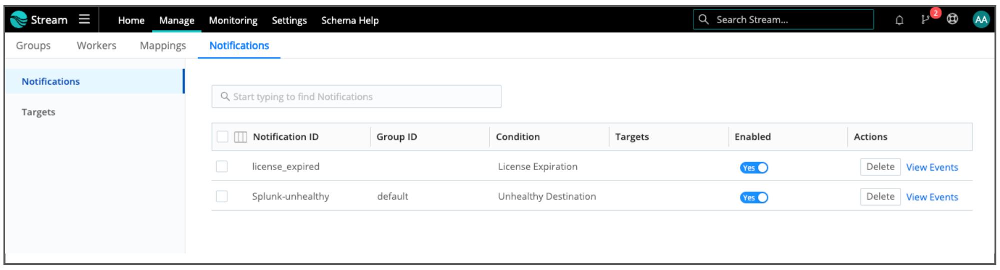
Once you've saved your Notifications, you can see Notification events specific to this Destination on the Destination config modal's **Events** tab. (When you set Source-state Notifications, a corresponding **Events** tab is available on Sources' and Collectors' config modals.) For a comprehensive view of all Notification events, see the systemwide Events Tab.

Managing Notifications

You can manage existing Notifications and targets from **Manage** > **Notifications**. You can also reach this UI by clicking an existing Notification in a Source or Destination modal or in the **Global Settings** > **Licensing** UI.

Notifications Tab

This tab lists all your configured Source-state and Unhealthy Destination Notifications, across all integrations, along with any configured license-expiration Notifications. You can't create new Notifications here, but you can disable or delete existing Notifications. For any individual Notification, click **View Events** to see events that have triggered the Notification.



Groups Workers Mappings Notifications

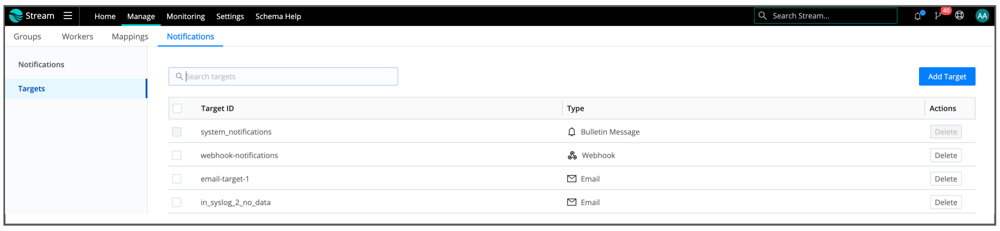
Notification ID	Group ID	Condition	Targets	Enabled	Actions
license_expired		License Expiration		Yes	Delete View Events
Splunk-unhealthy	default	Unhealthy Destination		Yes	Delete View Events

Notifications tab

To modify a Notification, click anywhere on its row.

Targets Tab

This tab is where you centrally configure and manage targets that are available across Cribl Stream – for all Sources, Destinations, and license-based Notifications. See Notification Targets for details.



Target ID	Type	Actions
system_notifications	Bulletin Message	Delete
webhook-notifications	Webhook	Delete
email-target-1	Email	Delete
in_syslog_2_no_data	Email	Delete

Targets tab

To create a new target, click **Add Target**. To delete a target, click **Delete** in the appropriate row.

9.5.1. Source-State Notifications

In Cribl Stream 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

Read on for details about these conditions and how to configure appropriate Notifications.

High Data Volume

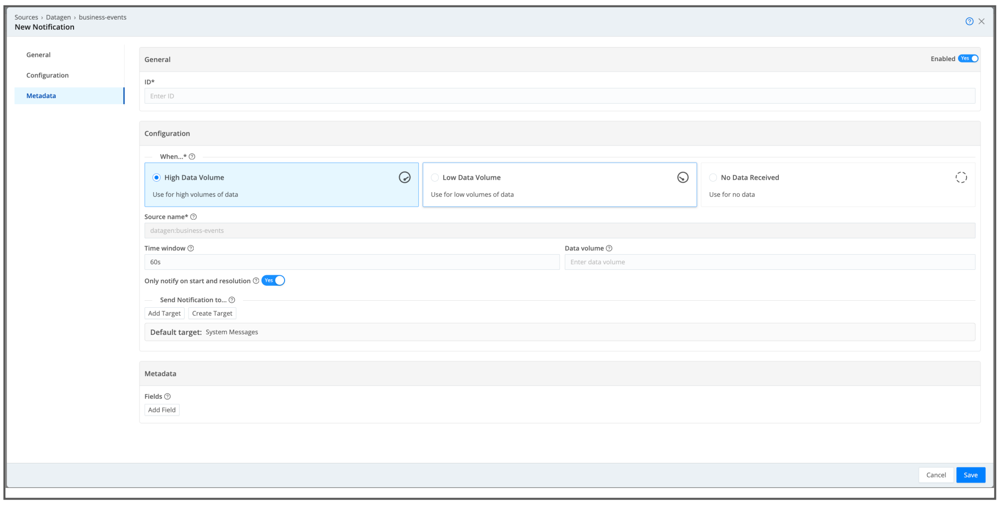
Cribl Stream will generate a Notification when incoming data over your configured **Time window** exceeds your configured **Data volume** threshold. This selection exposes the following fields:

Notification targets: The **Default target** is always locked to `System Messages`.

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

Time window: This field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when the Source has reported the trigger condition over the past 60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, and so forth.

Data volume: Enter the threshold above which a Notification will trigger. Accepts numerals with units like `KB`, `MB`, and so forth. For example: `4GB`.



Sources › Datagen › business-events
New Notification

General
Configuration
Metadata

General
Enabled: Yes
ID*: Enter ID

Configuration
When...*
[✓ High Data Volume] — Use for high volumes of data
Low Data Volume — Use for low volumes of data
No Data Received — Use for no data
Source name*: datagen:business-events
Time window: 60s
Data volume: Enter data volume
Only notify on start and resolution: Yes
Send Notification to...
Add Target | Create Target
Default target: System Messages

Metadata
Fields
Add Field

Cancel | Save

Configuring a high data volume Notification

Low Data Volume

Select the `Low Data Volume` tile to trigger Notifications when incoming data over your configured **Time window** is lower than your configured **Data volume** threshold.

This selection exposes the same additional fields as High Data Volume, except that here, the **Data volume** value defines a **floor** below which the Notification will trigger.

No Data

Select the `No Data Received` tile to trigger Notifications when the Source or Collector ingests zero data over your configured **Time window**.

This selection exposes the same additional fields as High Data Volume, except it omits the **Data volume** field. With no data entering the Source, there is no threshold to configure.

Source-State Notifications for Webhook Targets

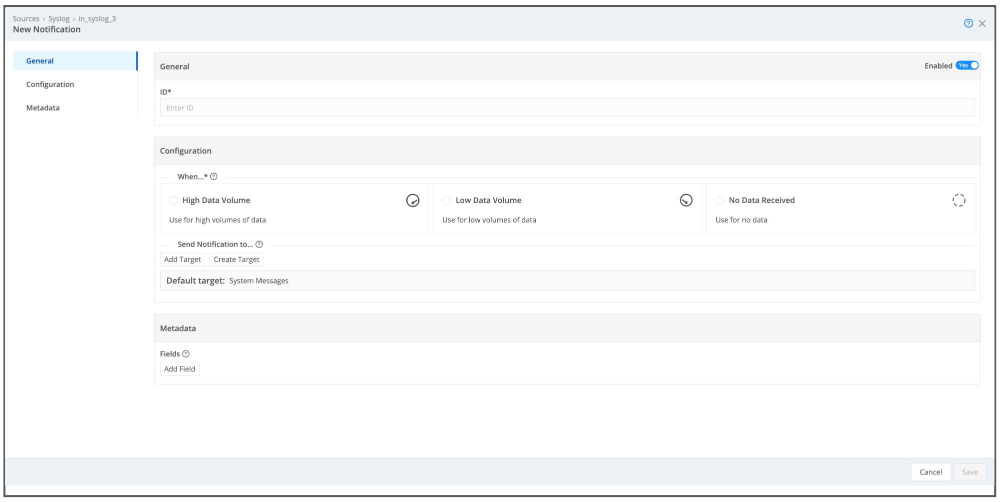
If you are sending a Source-state Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- Source High Data Volume
- Source Low Data Volume
- Source No Data Received

Configuring Source Notifications

To configure a Source-state Notification:

1. Configure and save the Source.
2. Access this Source's **Notifications** tab by one of the following methods:
    - Click the **Notifications** button on the **Manage...Destinations** page's appropriate row.
    - Reopen the Source's config modal and click its **Notifications** tab.
3. Click **Add Notification** to access the **New Notification** modal shown below.



Enabled: Yes

Default target: System Messages

Configuring a Source Notification

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

When: Select one of the following Notification tiles:

- **High Data Volume**
- **Low Data Volume**
- **No Data Received**

You can create multiple Notifications for the same Source, but you must configure them separately.

Send notification to: 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 Target** to configure a new target.

See Notification Targets for details.

Default target is always locked to `System Messages`.

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

Time window: This field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when a Destination or Source has reported the trigger condition over the past

60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, and so forth.

Only notify on start and resolution: When this option is set to `Yes`, Cribl Stream will send a Notification at the onset of the triggering condition. On resolution, it will send a second Notification.

If you don't enable this option and a Source-state Notification's trigger condition persists beyond your configured **Time window**, Cribl Stream will send a new Notification, once per **Time window** interval.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Stream includes in the notification payload. See Metadata for more information.

9.5.2. Destination-State Notifications

In Cribl Stream 3.5 and above, you can configure Notifications on Destinations that will trigger under these conditions:

- Destination Backpressure Activated
- Persistent Queue Usage
- Unhealthy Destination

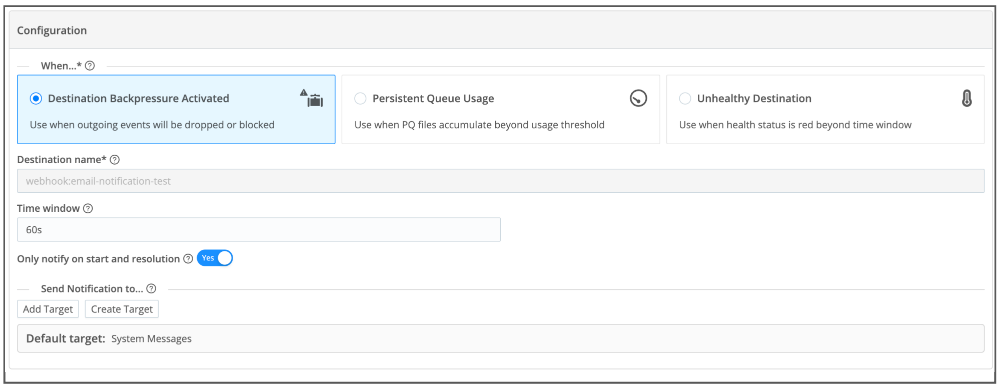
Read on for details about these conditions and how to configure appropriate Notifications.

Destination Backpressure Activated

Cribl Stream will generate a Notification when one of the following events occurs:

- The Destination's **Backpressure behavior** is set to `Block` or `Drop`, and backpressure causes outgoing events to block or drop.
- The Destination's **Backpressure behavior** is set to Persistent Queue, but its **Queue-full behavior** is set to either `Block` or `Drop new data`, and a filled queue causes the Destination to to block or drop outgoing events.

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


Configuration

When...* ⓘ

(●) Destination Backpressure Activated — Use when outgoing events will be dropped or blocked

( ) Persistent Queue Usage — Use when PQ files accumulate beyond usage threshold

( ) Unhealthy Destination — Use when health status is red beyond time window

Destination name* ⓘ: webhook:email-notification-test

Time window ⓘ: 60s

Only notify on start and resolution ⓘ: Yes

Send Notification to... ⓘ

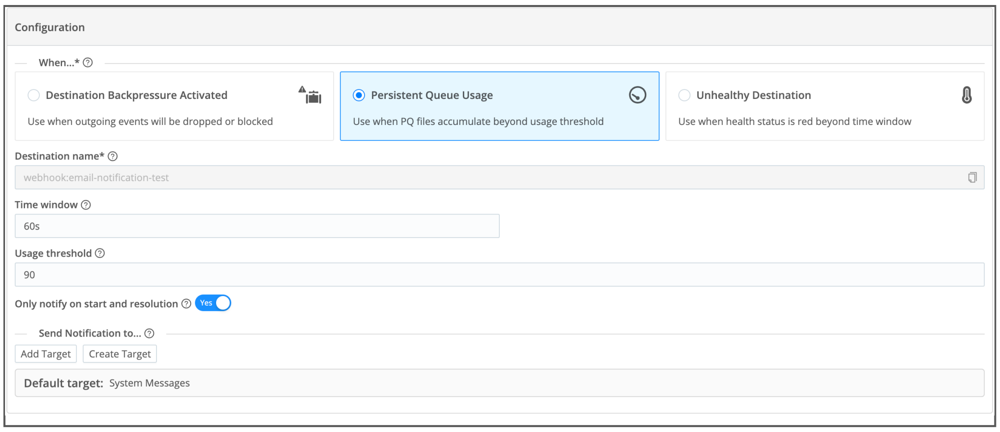
Add Target | Create Target

Default target: System Messages

Backpressure Notification

Persistent Queue Usage

Cribl Stream will generate a `Persistent Queue usage has surpassed <threshold>%` Notification when the PQ accumulates files past the `<threshold>` percentage of capacity that you set in the **Usage threshold** field. This field appears only when you configure a Notification for **Persistent Queue Usage**.


Configuration

When...*

Destination Backpressure Activated — Use when outgoing events will be dropped or blocked

(●) Persistent Queue Usage — Use when PQ files accumulate beyond usage threshold

Unhealthy Destination — Use when health status is red beyond time window

Destination name*: webhook:email-notification-test

Time window: 60s

Usage threshold: 90

Only notify on start and resolution: Yes

Send Notification to...

Add Target | Create Target

Default target: System Messages

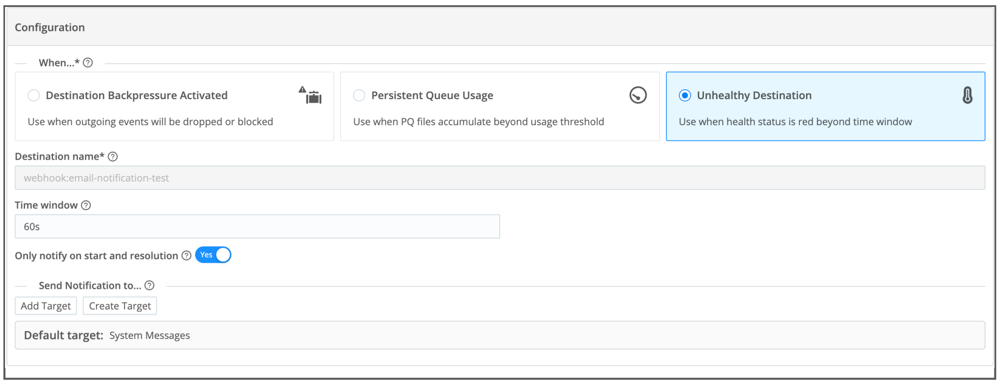
Persistent Queue Notification

Unhealthy Destination

Cribl Stream 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 Configuring Destination Notifications.

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

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


Configuration

When...*

Destination Backpressure Activated — Use when outgoing events will be dropped or blocked

Persistent Queue Usage — Use when PQ files accumulate beyond usage threshold

(●) Unhealthy Destination — Use when health status is red beyond time window

Destination name*: webhook:email-notification-test

Time window: 60s

Only notify on start and resolution: Yes

Send Notification to...

Add Target | Create Target

Default target: System Messages

Destination Unhealthy Notification

Destination-State Notifications for Webhook Targets

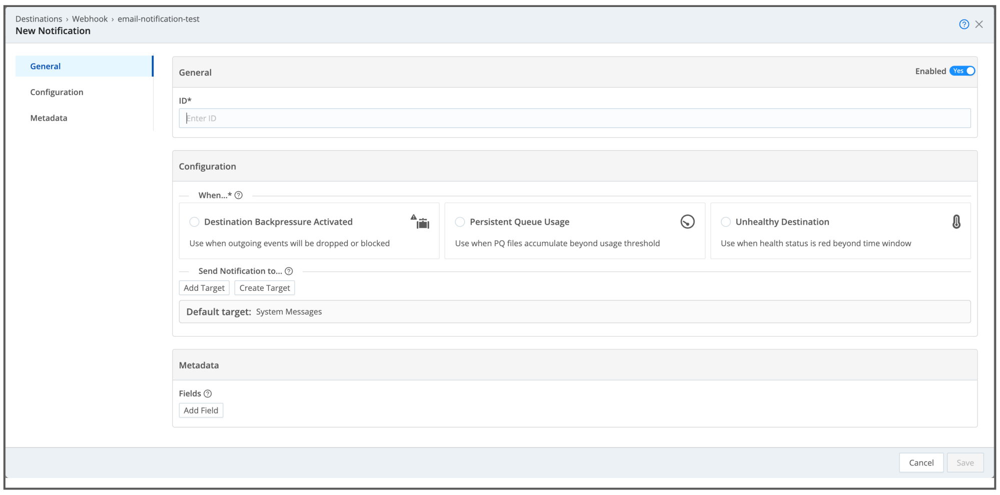
If you are sending a Destination-state Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- Unhealthy Destination
- Destination Backpressure Activated

Configuring Destination Notifications

To configure a Destination-state Notification:

1. Configure and save the Destination.
2. Access this Destination’s **Notifications** tab by one of the following methods:
    - 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.



General
Enabled: Yes
ID*: Enter ID

Configuration
When…*
Destination Backpressure Activated – Use when outgoing events will be dropped or blocked
Persistent Queue Usage – Use when PQ files accumulate beyond usage threshold
Unhealthy Destination – Use when health status is red beyond time window
Send Notification to…
Add Target | Create Target
Default target: System Messages

Metadata
Fields
Add Field

Configuring a Destination Notification

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

When: Select one of the following Notification tiles:

- **Destination Backpressure Activated**
- **Persistent Queue Usage**
- **Unhealthy Destination**

You can set up multiple Notifications for the same Destination, but you must configure them separately.

Send Notification to: 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 Target** to configure a new target.

See Notification Targets for details.

Default target is always locked to `System Messages`.

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

Time window: This field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when a Destination or Source has reported the trigger condition over the past 60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, and so forth.

Only notify on start and resolution: When this option is set to `Yes`, Cribl Stream will send a Notification at the onset of the triggering condition and a second Notification to report its resolution.

If you don't enable this option and a Destination-state Notification's trigger condition persists beyond your configured Time window, Cribl Stream will send a new Notification, once per **Time window** interval.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Stream includes in the notification payload. See Metadata for more information.

9.5.3. License-Expiration Notifications

To prevent interruptions in data throughput, you can configure a Notification that will be triggered two weeks before your Cribl Stream paid license expires, and then again upon expiration. (If the two-week Notification is cleared from the **Messages** tab between those dates, but the license has not been extended, it will trigger again.)

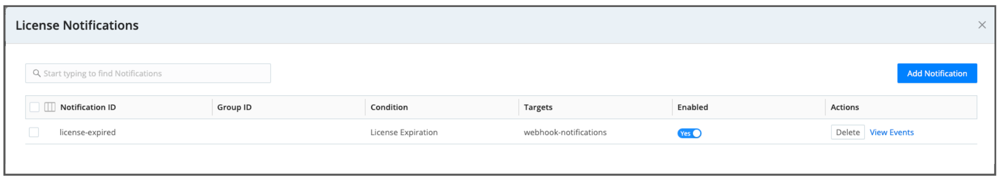
License-Expiration Notifications for Webhook Targets

If you are sending a license-expiration Notification to a Webhook Notification target, you can include a variety of expression fields in the target's **Source expression**. For more information, see:

- Fields Common to All Notification Types
- License Expiration

Configuring License-Expiration Notifications

1. From the top nav, select **Settings** > (**Global Settings** >) **Licensing**.
2. Click **Add Expiration Notification** to access the **License Notifications** modal.



Notification ID	Group ID	Condition	Targets	Enabled	Actions
license-expired		License Expiration	webhook-notifications	Yes	Delete View Events

License Notifications modal

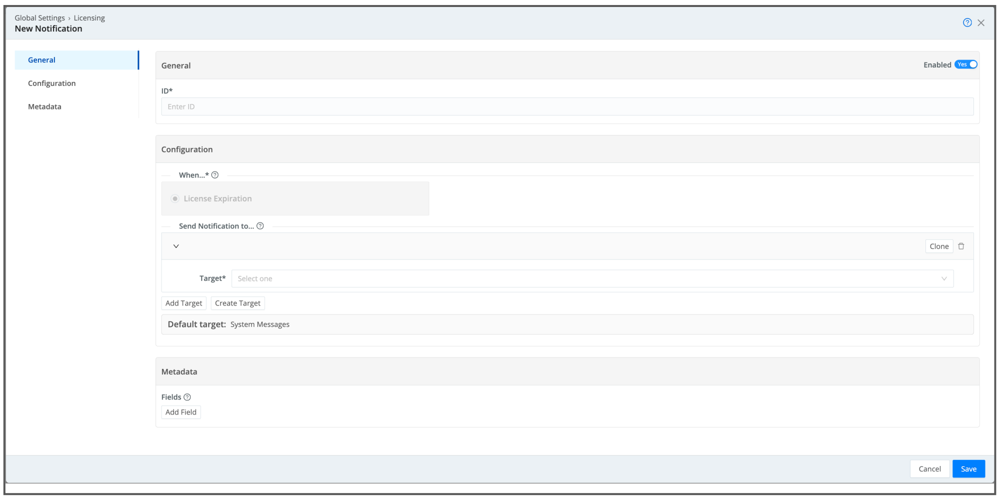
This modal shows existing license Notifications.

To modify a license-expiration Notification, click anywhere on its row.

To delete a Notification, click **Delete**.

To view license-expiration events, click **View Events**.

Click **Add Notification** to to access the **New Notification** modal shown below.



Configuring a license expiration Notification

This **New Notification** modal provides General, Configuration, and Metadata tabs.

General

ID: Enter a unique ID for this Notification. Notifications are enabled by default, but you can disable the Notification by setting **Enabled** to `No`.

Configuration

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

Send notification to: This section contains a list of the targets receiving this Notification. The **Default target** is always locked to `System Messages`.

Click **Add Target** for each additional existing target that you want to send this Notification to. Click **Create Target** to create a new target for the Notification.

Metadata

You can enter user-defined fields called *metadata*, which Cribl Stream includes in the notification payload. See Metadata for more information.

9.5.4. Email Notifications

If you're a Cribl Stream admin, email notifications make it easy to receive alerts about any operational issues that require your attention, such as a particular Source or Destination condition or a pending license expiration.

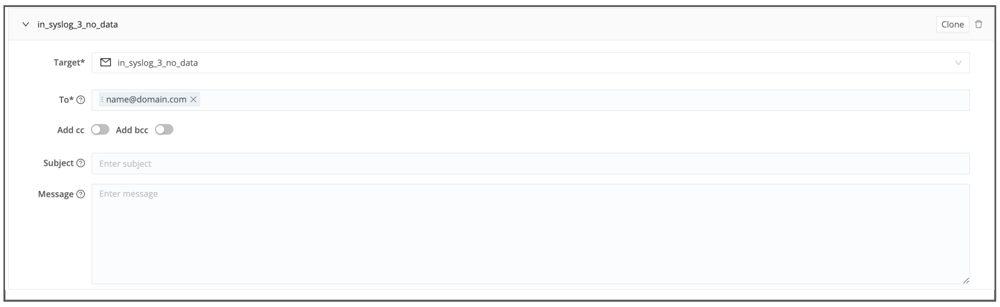
An email Notification requires two things — a configured Notification and an email Notification target. Cribl.Cloud organizations using an Enterprise license also have access to a preconfigured email Notification target.

Email Notifications do not have an unsubscribe option. Recipients who do not want to receive particular email Notifications should contact their Cribl Stream admins.

Email Notifications are sent on a no-reply basis.

Configuring Email Notifications

When you create a Notification for an email target, specify the recipients of the message, the subject line, and the contents of the message.



in_syslog_3_no_data

Target*	in_syslog_3_no_data
To*	name@domain.com
Add cc	Add bcc
Subject	Enter subject
Message	Enter message

Email Notification with target selected

Target: The ID of the email target to which you want to send the Notification.

Click **Add target** to add an existing target.

Click **Create target** to create a new target. For information on configuring email Notification targets, see Email Notification Targets.

If the Notification already has a designated target, you can change the selection by clicking the drop-down.

When an email notification target is selected, the following additional fields appear:

To: The email address of the recipient.

Add cc: When enabled, reveals a field where you can enter the addresses of additional recipients.

Add bcc: When enabled, reveals a field where you can enter the addresses of additional recipients that do not appear in the Notification email.

Cribl Stream does not limit the total number of recipients for a Notification, but your email service might set a limit.

Subject: The subject line of the email Notification. You can use variables in the subject line.

Message: The content of an email Notification. You can use variables in the body of the email message.

Email Notification Variables

Email variables are placeholders in the email template that get replaced with actual values when the email is sent. These variables can be:

- General-use variables, like `condition`, `worker_group`, or `timestamp`
- Special-purpose variables

You can use a variety of general-use and special-purpose variables in the subject or message body of an email Notification. Insert a variable name between two braces preceded by a `$`. For example: `${cribl_notification}`.

General-Use Email Variables

Variable	Description
`workspace`	Workspace name (Cribl.Cloud only).
`organization`	Organization ID (Cribl.Cloud only).
`timestamp`	Timestamp when the email is sent. For example: `2019-08-04 18:22:24 UTC`.
`cribl_notification`	User-defined notification ID.

Special-Purpose Email Variables

Variable	Usage	Example
`input`	Type and name of a Source or Collector.	`syslog:in_syslog_1`
`output`	Type and name of a Destination.	`webhook:out_webhook_1`
`name`		Unique ID of a Source or Destination.
`bytes`	Quantity of data triggering the specified Notification.	`0`
`starttime`	Start time of an Event (in Epoch seconds).	`1706747185`
`endtime`	End time of an Event (in Epoch seconds).	`1706747294`
`health`	Health metric of the specified Source or Destination.	`0`
`_raw`	The `_raw` field of the Event triggering the Notification.	`_raw (Source ${name} in group ${__workerGroup} traffic volume greater than ${dataVolume} in ${timeWindow})`
`backpressure_type`	Backpressure behavior.	`1=BLOCKING, 2=DROPPING`
`queue_usage`	Percentage of capacity set in the **Usage threshold** field for a **Persistent Queue Usage** Notification.	`90`

Troubleshooting Email Notifications

The section details the troubleshooting steps you can take if an email notification fails to reach its intended recipient.

Test the Email Notification Target

An email notification can fail if the target is misconfigured. You can test your email notification target by following this procedure:

1. Open the target (in **Manage** > **Notifications** > **Targets**).

2. Click **Test Target**.
3. Add one or more email addresses to the **Test Target** modal and click **Send Test Email**.
4. Check the designated inbox to verify receipt of the test message. If it does not arrive in the designated inbox, review the target configuration.

Check Notification Service Logs

A failed email notification leaves a log entry. You can examine logs by following this procedure:

1. Select **Monitoring** > **Logs**.
2. Open the **Logs** drop-down and select **Leader** > **Notifications Service**.
3. Examine any logs that have errors.

Check Notifications

Cribl Stream stores Notifications. You can check them by following this procedure:

1. Select **Monitoring** > **Notifications**.
2. Select the **cribl_notification** field.
3. Search for **cribl_notification** fields corresponding to the Notification ID of the failed Notification.

9.5.5. Notification Targets

A Notification target specifies the delivery method for a Notification. Every Notification requires one or more targets.

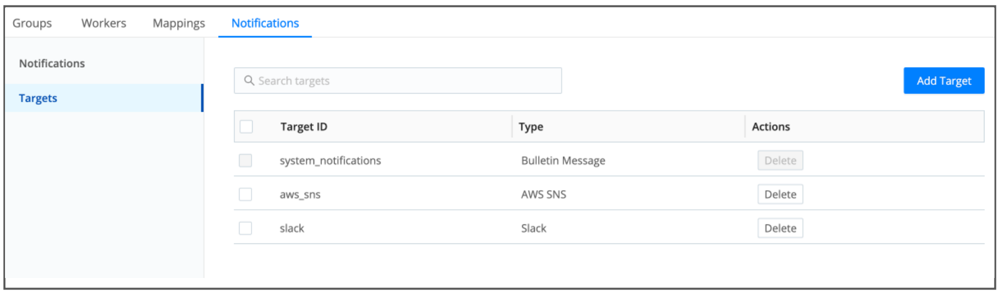
Available target types include:

- Webhook
- PagerDuty integration
- Slack
- AWS SNS
- Email

Adding a Notification Target

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. Select the desired **Target type**.



Target ID	Type	Actions
system_notifications	Bulletin Message	Delete
aws_sns	AWS SNS	Delete
slack	Slack	Delete

Add targets on the **Targets** tab

Then configure the target as described in the appropriate page:

- Webhook
- PagerDuty
- Slack
- AWS SNS
- Email

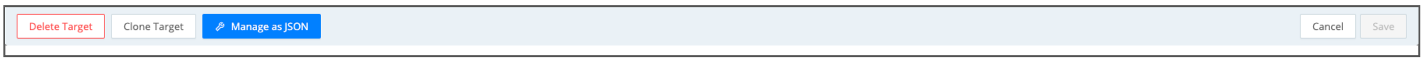
When you create a Notification target in Cribl Stream, Edge, or Search, it will also be available to you in the other products. For example, if you create a Notification target in Cribl Stream, you can also access it in Cribl Edge and Cribl Search.

Notifications require an Enterprise or Standard license. Without an appropriate license, target configuration options will be hidden or disabled in Cribl Stream's UI.

Managing Notification Targets

You can manage targets by selecting **Manage** > **Notifications** > **Targets**.

Click any existing Notification target to open a modal where you can manage the target, using buttons at the bottom of the UI.



Manage notification targets

Click **Delete Target** to delete an existing Notification target. You can also select multiple targets for deletion, using the check boxes on the left side of the modal.

Click **Clone Target** to copy an existing target. A modal will open so you can modify the configuration of the target you started with.

To edit any target's definition in a JSON text editor, click **Manage as JSON** at the bottom of the **Notifications** > **Targets** modal. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to copy and modify existing target configurations as `.json` files.

Some targets have additional controls for target configuration. See the documentation for the target of interest for more information.

9.5.5.1. Webhook Notification Targets

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

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can’t change it later, so make sure you like it.

Configuration

The added options that appear on this first left tab are:

URL: The endpoint that should receive Cribl Stream 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 Stream sends to the endpoint – for example: `notification=${_raw}`. For other fields you can use, see Expression Fields. If empty, Cribl Stream will send the full notification event as stringified JSON.
    - **Drop when null**: Toggle to `Yes` if you want to drop events where the above **Source expression** evaluates to `null`.
    - **Event delimiter**: Delimiter string to insert between individual events. Defaults to newline character (`\n`).
    - **Content type**: Defaults to `application/x-ndjson`. You can substitute a different content type for requests sent to the endpoint. This entry will be overridden by any content types set in this modal’s **Advanced Settings** tab > **Extra HTTP Headers** section.

   - **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${events}` variable. An example expression to send the batch inside a JSON object would be: `{"items" : [${events}] }`.

Expression Fields

When building the Source expression, you can use the following fields:

Fields Common to All Notification Types

- `starttime`: Beginning of the time bucket where this condition was reported. All Notifications have this field.
- `endtime`: End of the time bucket where this condition was reported. All Notifications have this field.
- `_time`: Timestamp when this Notification was created. All Notifications have this.
- `cribl_host`: Hostname of the (physical or virtual) machine on which this Notification was created. All Destination and Source Notifications have this.
- `cribl_notification`: Configured name/ID of this Notification. All Destination and Source Notifications have this.
- `origin_metadata`: Object containing metadata about the Notification's origin, with the following fields for all Destination Notifications:
   - `type`: "output".
   - `id`: ID of the affected Destination.
   - `subType`: Destination's type (where applicable).
- `origin_metadata`: Object containing metadata about the Notification's origin, with the following fields for all Source Notifications:
   - `type`: "input".
   - `id`: ID of the affected Source.
   - `subType`: Source's type (where applicable).

Unhealthy Destination

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `output`: Output ID of the affected Destination.
- `_raw`: "Destination `${output}` [in group `${__worker_group}`] is unhealthy".
- `_metric`: "health.outputs".

Destination Persistent Queue Usage

- `output`: Output ID of the affected Destination.
- `timeWindow`: The interval at which notifications will be repeated.
- `usageThreshold`: The minimum persistent queue utilization (stated as a percentage) that will trigger Notifications.
- `_metric`: “system.pq_used”.
- `usage`: The persistent queue utilization (stated as a percentage) that triggered the Notification.

Destination Backpressure Activated

- `backpressure_type`: `1` for Block, `2` for Drop.
- `output`: Output ID of the affected Destination.
- `_raw`: “Backpressure ([dropping|blocking]) is engaged for destination `${output}` [in group `${__worker_group}`]”.
- `_metric`: “backpressure.outputs”.

Source High Data Volume

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `bytes`: Number of bytes received in the time bucket.
- `input`: Input ID of the affected Source.
- `_raw`: “Source `${input}` [in group `${__worker_group}`] traffic volume greater than `${dataVolume}` in `${timeWindow}`”.
- `_metric`: “total.in_bytes”.

Source Low Data Volume

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `bytes`: Number of bytes received in the time bucket.
- `input`: Input ID of the affected Source.
- `_raw`: “Source `${input}` [in group `${__worker_group}`] traffic volume less than `${dataVolume}` in `${timeWindow}`”.
- `_metric`: “total.in_bytes”.

Source No Data Received

- `health`: Numeric value where `0`=green, `1`=yellow, `2`=red.
- `_time`: Timestamp when this Notification was created.
- `input`: Input ID of the affected Source.
- `_raw`: "Source `${input}` [in group `${__worker_group}`] had no data for `${timeWindow}`".
- `_metric`: "total.in_bytes"

License Expiration

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

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Basic**: In the resulting **Username** and **Password** fields, enter HTTP Basic authentication credentials.

Post-Processing Settings

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates not authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Keep alive: By default, Cribl Stream sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Stream to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB, but you can set this limit to as high as 500 MB (512000 KB).

High values can cause high memory usage per Worker Node, especially if a dynamically constructed URL causes this target to send events to more than one URL. The actual request body size might exceed the specified value because the target adds bytes when it writes to the downstream receiver. Therefore, we recommend that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field; headers added by this method take precedence over headers defined in the **Extra HTTP headers** table.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Used to declare headers that are safe to log as plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

9.5.5.2. PagerDuty Notification Targets

This option sends Cribl Stream Notifications to PagerDuty, a real-time incident response platform, using Cribl Stream's native integration with the PagerDuty API. Select Select **PagerDuty** in the **New Target** modal to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Routing key: Enter your 32-character Integration key on a PagerDuty service or global ruleset.

Group: Optionally, specify a PagerDuty default group to assign to Cribl Stream Notifications.

Class: Optionally, specify a PagerDuty default class to assign to Cribl Stream Notifications.

Component: Optionally, a PagerDuty default component value to assign to Cribl Stream Notification. (This field is prefilled with `logstream`.)

Severity: Set the default message severity for events sent to PagerDuty. Defaults to `info`; you can instead select `error`, `warning`, or `critical`. (Will be overridden by the `__severity` value, if set.)

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

9.5.5.3. Slack Notification Targets

You can send notifications to a Slack channel, using Slack's Incoming Webhooks.

> You can also use a Webhook Destination to send specific event data to Slack for a team's attention. For more information, see our Slack/Webhook integration topic.

First, in your Slack workspace, use a Slack app to enable incoming webhooks.

Then, in the **New Target** modal, click **Slack** to expose the following additional options on the modal's (single) **General Settings** left tab:

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. You can't change it later, so make sure you like it.

Configuration

Webhook URL: Add the full URL of your Slack Incoming Webhook. For example: `https://hooks.slack.com/services/T00000000/B00000000/XXXXXXXXXXXXXXXXXXXXXXXX`.

Retries

Honor Retry-After header: When toggled to `Yes` and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No` (the default), Cribl Stream ignores all `retry-after` headers.

Settings for failed HTTP requests: Automatically retries after unsuccessful response status codes, such as `429` (Too Many Requests) or `503` (Service Unavailable). Clicking **Add Setting** reveals a table where you can add retry parameters for individual failed HTTP requests. These include:

- **HTTP status code**: The individual status code for which the retry parameters apply.

- **Pre-backoff interval (ms)**: How long, in milliseconds, Cribl Stream should wait before initiating backoff. The maximum interval is 600,000 ms (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Cribl Stream should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

Retry timed-out HTTP requests: When set to `Yes`, Cribl Stream to automatically retries HTTP requests that have timed out. Defaults to `No`. Enabling this option exposes the following settings:

- **Pre-backoff interval (ms)**: How long, in milliseconds, Cribl Stream should wait before initiating backoff. Maximum interval is `600,000 ms` (10 minutes).
- **Backoff multiplier**: Base for exponential backoff. A value of `2` (default) means Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.
- **Backoff limit (ms)**: The maximum backoff interval, in milliseconds, Cribl Stream should apply. Default (and minimum) is 10,000 ms (10 seconds); maximum is 180,000 ms (180 seconds).

9.5.5.4. AWS SNS Notification Targets

You can send notifications to an Amazon Simple Notification Service (SNS) topic. This gives you access to a broad array of notification destinations, such as various AWS services, mobile push notifications, or text messages.

To add an Amazon SNS Notification target in Cribl Stream, go to **Manage** > **Notifications** > **Targets** > **Add Target**.

General Settings

Target ID: Enter a unique ID used to identify the target. This will show in the **Target ID** column of the **Targets** tab. It can't be changed later, so make sure you like it.

Configuration

Destination type: Defaults to **Topic ARN**. The SMS section below explains the **Phone number** option.

Region: Select the region associated with the Amazon S3 bucket.

Default Topic ARN: The default Amazon Resource Name (ARN) of the Amazon SNS topic to which you want to send notifications. Cribl Stream expects the ARN in a format like this:

`arn:aws:sns:region:account-id:MyTopic`.

If you use a non-AWS URL, the format must be:

`{url}/myQueueName` – for example, `https://host:port/myQueueName`.

Must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at initialization time. For example, if you're referencing a Global Variable: `https://host:port/myQueue-${C.vars.myVar}`. This value can be overridden by the notification event `__topicArn` field.

Phone number allowlist: A wildcard list of phone numbers that are allowed to receive SMS notifications. This is used when **Destination type** is set to **Phone number**.

Authentication

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance, local credentials, sidecar, or other source. The attached IAM role grants Cribl access to authorized AWS resources. Can also use the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`. Works only when running on AWS.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Worker Nodes not in an AWS VPC, like those running a private cloud.

The **Manual** option exposes these additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to the `env.AWS_ACCESS_KEY_ID` environment variable, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to the `env.AWS_SECRET_ACCESS_KEY` environment variable, or to the metadata endpoint for IAM credentials.

The values for **Access key** and **Secret key** can be a constant, or a JavaScript expression (such as `${C.env.MY_VAR}`) enclosed in quotes or backticks, which allows configuration with environment variables.

Secret: If not running on AWS, you can select this option to supply a stored secret that references an AWS access key and secret key. The Secret option exposes this additional field:

- **Secret key pair**: Use the drop-down to select an API key/secret key pair that you've configured in Cribl Stream's secrets manager. To store a new, reusable secret, click **Create**.

Assume Role

Enable for SNS: Toggle to `Yes` to define an IAM Role to use, instead of automatically detecting one locally.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Post-Processing

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Advanced Settings

Maximum number of retries: The maximum number of retries before the output returns an error. The retries use an exponential backoff policy.

Endpoint: The SNS service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to an SNS-compatible endpoint.

Signature version: Signature version to use for signing SNS requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to `Yes`.

SMS Notifications

You can use an Amazon SNS Notification target to send text messages (SMS) to a list of phone numbers. To do this, you'll need to set up a whitelist of phone numbers that are allowed to receive notifications.

1. Go to **Manage** > **Notifications** > **Targets** > **Add Target**.
2. Enter a unique **Target ID**.
3. Set the **Target type** to **AWS SNS**.
4. Set **Destination type** to **Phone number**.
5. Set **Region** to the region of the Amazon S3 bucket.
6. In **Default Phone number**, enter a comma-separated list of phone numbers that are allowed to receive notifications. This value can be overridden by the notification event `__phoneNumber` field. You can use `*` as the wildcard character.
   For example: `+15555550123, +15555551***`.
7. If desired, use **Phone number allowlist** to specify a wildcard list of allowed phone numbers.
8. Configure the remaining sections of the AWS SNS Notification target as described above.

9. Select **Save**.

Now, when you set up Notifications, you can select the new Amazon SNS target and specify any phone number that matches the configured whitelist.

9.5.5.5. Email Notification Targets

You can send notifications by email, using an SMTP server of your choice. Once you have configured the email notification target, you can specify the recipients and customize the subject line and message content. See Email Notifications for details on configuring an email Notification.

To add an email Notification target in Cribl Stream, go to **Manage** > **Notifications** > **Targets** > **Add Target**.

If you are a Cribl.Cloud user with an Enterprise license, you can use the default email Notification target. This target requires no configuration.

General Settings

Target ID: Enter a unique name to identify this email Notification target.

Configuration

Address: Identify the SMTP server by its hostname or IP address.

Port: Set the SMTP port. Use port `587` for SMTP Secure (SMTPS). You can also use port `25`. Use `465` when SSL/TLS is enabled. You can also use port `2525` if your email service provider supports this port as a backup when other ports are blocked by a network provider or a firewall.

Encryption type: Specify the encryption type used to secure SMTP communication. Options include:

- **STARTTLS**: Select this option to start the connection as plaintext, then upgrade it to a TLS-encrypted one if the server supports it. If the server doesn't support it, the connection remains plaintext.

  > **STARTTLS** upgrades the connection to be encrypted but does not authenticate the server. Take additional steps to prevent man-in-the-middle attacks.

- **Require STARTTLS**: Select this option to require TLS. If the server doesn't support a secure connection, the connection will be dropped.
- **TLS (SMTPS)**: Select this option to use an encrypted connection from the start without requiring a subsequent connection upgrade.
- **None**: Select this option to use a plaintext connection.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path** or by another trusted CA (such as the system's CA).

From: Identify the email address of the sender.

Authentication

Username: The authentication principal (if required).

Password: The authentication credential (if required).

Testing Email Notification Targets

When you finish configuring your email Notification target, you can manage it, using the following options.

Click **Save & Test Target** to save your target and open a modal where you can enter an email to test your email Notification target. After configuration, the button text changes to **Test Target**.

Default Email Notification Target for Cribl.Cloud Enterprise Orgs

For Cribl.Cloud Enterprise orgs, a default email Notification target is available for Cribl Stream, Edge, and Search. You can use this target to route any email Notification to any valid email address.

This target appears in the **Notifications** > **Target** modal as `system_email`. You cannot modify or remove this target.

The `system_email` Notification target is managed by Cribl. It will be disabled in the unlikely event of abuse. If this target is disabled for a workspace, a log entry will appear in the Notifications Service logs. Select **Monitoring** > **Logs** > **Notifications Service** to view this log.

The `system_email` email Notification target is available only for Cribl.Cloud Enterprise organizations.

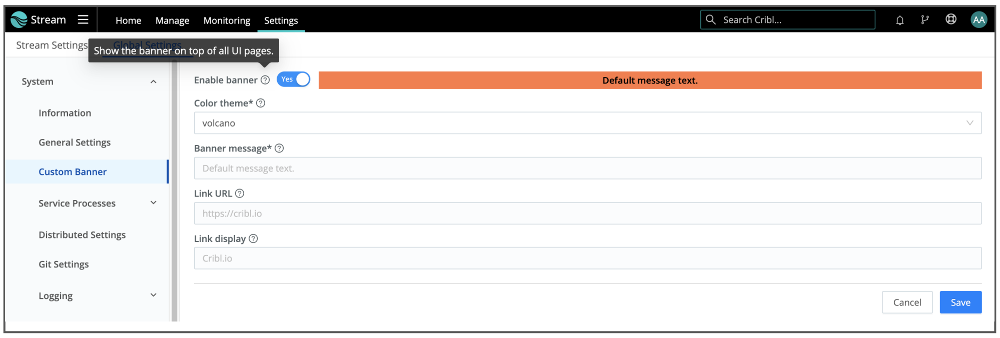
Sending Domain for Cribl.Cloud Email Notifications

Every Cribl.Cloud workspace and organization has a unique address for its email Notification target. Messages sent using this target will have a sender's address in the form `do-not-reply@<workspace>-<org>.criblcloud.email`. To ensure delivery, recipients should add the `criblcloud.email` domain to their email allowlists.

9.6. Custom Banners

In version 4.3 and later, Cribl admins can define a custom banner from the Leader. This will be displayed at the top of your Cribl Organization's UI, above all Cribl apps. You can display one banner at a time.

Your banner can display warnings, requirements, neutral updates, or good news about new features or options. It can contain a mixture of text, a raw URL, or a URL with a friendly/short label. The banner does not appear unless you enable and configure it, as outlined below.



Enable banner: Yes

Default message text.

Color theme*: volcano

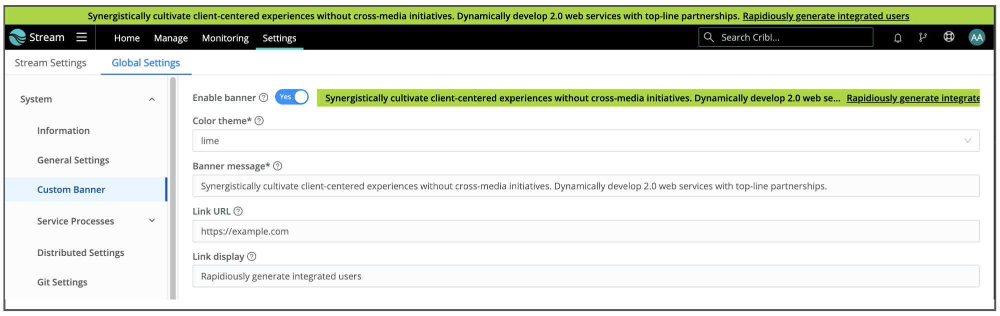
Banner message*: Default message text.

Link URL: https://cribl.io

Link display: Cribl.io

Configuring/enabling a custom banner

1. From Cribl Stream's top nav, select **Settings** > **Global Settings**.
2. From the resulting left nav, select **System** > **Custom Banner**.
3. Toggle the **Enable banner** slider on. This exposes the configuration options below.
4. From the **Color theme** drop-down, select a background color matching your banner's severity. (Cribl reserves dark red, light blue, and light green for system notifications, so the drop-down offers alternative shades of these colors.)
5. In the **Banner message** field, enter a single-line message of up to 1,024 characters. (The displayed banner will truncate characters that cannot fit across one line of a given viewport.)
6. In the **Link URL** field, optionally enter a hyperlink to append to the message.
7. In the **Link display** field, optionally give your **Link URL** a text label. This can include up to 512 characters, but a long label can force-truncate the **Banner message**. A short label can help everything fit.
8. Click **Save** to display your banner.
9. Commit your changes from the Leader.


Synergistically cultivate client-centered experiences without cross-media initiatives. Dynamically develop 2.0 web services with top-line partnerships. Rapidiously generate integrated users

Enable banner: Yes

Color theme*: lime

Banner message*: Synergistically cultivate client-centered experiences without cross-media initiatives. Dynamically develop 2.0 web services with top-line partnerships.

Link URL: https://example.com

Link display: Rapidiously generate integrated users

Custom banner, configured and displayed

To hide the banner, toggle the **Enable banner** slider off, and again **Save** and commit your changes. Your banner's configuration will remain intact, ready to re-enable and/or revise when you're ready.

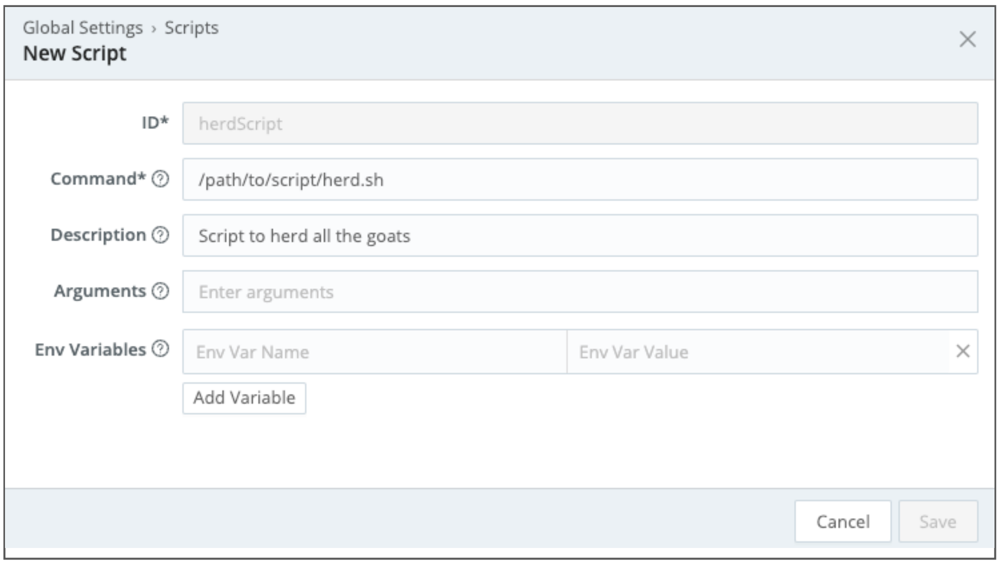
9.7. Scripts

Admins can run scripts (for example, shell scripts) from within Cribl Stream 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 Stream is running. Make sure you understand the impact of what you're executing before you do so!

To add a new script, click **Add script** on the scripts page.


Global Settings › Scripts
New Script

ID*	herdScript
Command*	/path/to/script/herd.sh
Description	Script to herd all the goats
Arguments	Enter arguments
Env Variables	Env Var Name / Env Var Value

Settings > Scripts page

The **New Script** modal provides the following fields:

- **ID**: Unique ID for this script.
- **Command**: Command to execute for this script.
- **Description**: Brief description about this script. Optional.
- **Arguments**: Arguments to pass when executing this script.
- **Env variables**: Extra environment variables to set when executing script.

Scripts in Distributed Deployments

- Scripts can be deployed from Leader Node, but can be **run** only locally from each Worker Node.

> - If the Script command is referencing a file (for example, `herd.sh`), that file must exist on the Cribl Stream instance. In other words, the Script management interface cannot be used to upload or manage script files.

9.8. Upgrading

This page outlines how you can upgrade a Cribl Stream single-instance or distributed deployment along one of these 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 Worker Nodes directly from the command line, you don't have to worry about a version upgrade path - there are no restrictions on the versions you can upgrade from or to.

Considerations

Here are some considerations to look at before you upgrade.

Upgrading to v.4.3.x using Leader-Managed Upgrades for Worker Nodes

If a Leader is on Cribl Stream 4.3.x, you should upgrade all Worker Nodes to 4.3.x **before** making any config changes or committing and deploying to the Nodes.

If you have already committed and deployed to Nodes that are on a version earlier than 4.3.0, and the Leader is on 4.3.x, you won't be able to upgrade the Nodes. In this case, here are three workaround options:

- Revert and redeploy the last commit, then upgrade the Worker Nodes and deploy the most up-to-date changes; **or**
- Upgrade the Worker Nodes via command line or script; **or**
- Downgrade the Leader to v.4.2.2, and wait for a future release that will enable you to upgrade Worker Nodes across all Leaders and 4.x.x versions.

If you run into issues, contact support@cribl.io for resolution help.

General Upgrade Considerations

Cribl Stream 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.

Cribl.Cloud Upgrades

Cribl.Cloud automatically upgrades the Leader Node when a new version is released. If you have hybrid (customer-managed) Cribl.Cloud Worker Groups, you need to upgrade them manually.

We recommend keeping the Leader and Worker Groups on the same version for maximum compatibility. If the versions differ, the following setups are supported:

- **Minor versions**: A Worker Group that is one minor version behind the Leader Node (for example: Worker Group on version v4.5.1 and Leader on v4.6.0) **is supported**. However, features introduced in the new version will not function on the Worker Group.
  A greater version difference may cause compatibility issues and is **not supported**.
- **Patch versions**: Leaders and Worker Groups on different patch versions within the same minor version (for example: Worker Group on v4.5.0 and Leader on v4.5.1) **are supported**.

Requirements for Specific Version Upgrades

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

Safeguarding Unencrypted Private Key Files for Rollback

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.

Standalone/Single-Instance

This path requires upgrading only the single/standalone node:

1. Stop Cribl Stream.
2. Uncompress the new version on top of the old one.
   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.
3. Restart Cribl Stream.

Distributed Deployment

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

First, upgrade the Leader Node. Then, upgrade the Worker Nodes. Lastly, commit and deploy the changes on the Leader.

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

1. Stop both Leaders.

> 2. Upgrade the primary Leader.
> 3. Upgrade the second Leader.
> 4. Upgrade each Worker Node, respectively.

Upgrade the Leader Node

1. Commit and deploy your desired previous version. (This will be your most recent checkpoint.) Optionally `git push` to your configured remote repo.
2. Stop Cribl Stream.
   - Optional but recommended: Back up the entire `$CRIBL_HOME` directory.
   - Optional: Check that the Worker Nodes are still functioning as expected. In the absence of the Leader Node, they should continue to work with their last deployed configurations.
3. Uncompress the new Cribl Stream version on top of the old one.
4. Change ownership of the `cribl` directory as needed: `[sudo] chown -R cribl:cribl $CRIBL_HOME`
5. Restart Cribl Stream and log back in.
6. Wait for all the Worker Nodes to report to the Leader, and ensure that they are correctly reporting the last committed configuration version.

Upgrading from 4.1.0

When upgrading a Leader Node from version 4.1.0 using the UI, you may encounter the following error:

```
{"status":"error","message":"Protocol \"https:\" not supported. Expected
\"http:\"","error":{"code":"ERR_INVALID_PROTOCOL"}}
```

To resolve this, configure the `no_proxy` directive to include the API's listening address on the Leader Node. To encompass all addresses, set `no_proxy` to `0.0.0.0`. This prevents internal traffic on the Leader Node from passing through the proxy server.

Here's an example `no-proxy` configuration:

```
http_proxy=http://localhost:1234 https_proxy=http://localhost:1234
no_proxy=0.0.0.0 bin/cribl start
```

This extra step is due to changes in communication between the connection listener service and the API server via an `http` client.

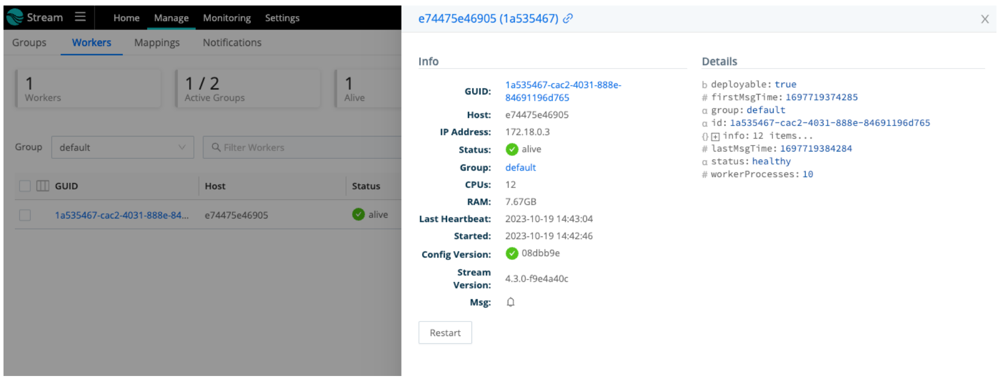
Upgrade the Worker Nodes

These are the same basic steps as when upgrading a single instance deployment, above:

1. Stop Cribl Stream on each Worker Node.
2. Uncompress the new version on top of the old one.
3. Change ownership of the `cribl` directory as needed: `[sudo] chown -R cribl:cribl $CRIBL_HOME`
4. Restart Cribl Stream.

Commit and Deploy All Changes from the Leader Node

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



GUID	Host	Status
1a535467-cac2-4031-888e-84...	e74475e46905	alive

e74475e46905 (1a535467)

Info

GUID:	1a535467-cac2-4031-888e-84691196d765
Host:	e74475e46905
IP Address:	172.18.0.3
Status:	alive
Group:	default
CPUs:	12
RAM:	7.67GB
Last Heartbeat:	2023-10-19 14:43:04
Started:	2023-10-19 14:42:46
Config Version:	08dbb9e
Stream Version:	4.3.0-f9e4a40c
Msg:	

Details

```
b deployable: true
# firstMsgTime: 1697719374285
a group: default
a id: 1a535467-cac2-4031-888e-84691196d765
{} [+] info: 12 items...
# lastMsgTime: 1697719384284
a status: healthy
# workerProcesses: 10
```

Successful upgrade confirmation

Breaking Change at v.3.2

Because of a breaking change at v.3.2, Leaders running v.3.2.x cannot upgrade (via the UI) Workers running versions prior to 3.2.0. The upgrade will fail with errors of the form: `Error checking upgrade path` and `Cannot read property 'greaterThan' of undefined`.

The workaround is to upgrade these Workers via the filesystem to v.3.2.0 or higher. The error does not affect upgrades of Workers running v.3.2.0+.

Accessing Upgrades via the UI

Starting in Cribl Stream v.2.4.4, you can upgrade using the UI. When you choose to upgrade via the UI, Cribl Stream stops its own server, updates the installed package, and restarts Stream for you.

Here's how to access the Upgrade UI in Stream:

1. First, select **Settings** from the top navigation.
2. Select **Upgrade** depending on your deployment type:
   - In a single-instance deployment, select **System**, then **Upgrade**.
   - To upgrade the Leader Node, select **Global Settings**, then **System** and **Upgrade**.
   - To upgrade a single Worker Group, select **Stream Settings** then **Group Upgrade**.

You can also enable an automatic backup and rollback in case an upgrade fails.

These options will work only if all Stream instances (including Worker Processes) start at v.2.4.4 or higher. For other version-specific limitations, please see Known Issues.

Upgrading the Leader via the UI

To upgrade the Leader in a distributed deployment, go to **Settings**, then select **Upgrade** under **Global Settings**.

When upgrading, you can choose to either download upgrade packages from Cribl's content delivery network (CDN), or from a filesystem location that you specify in the form of a path.

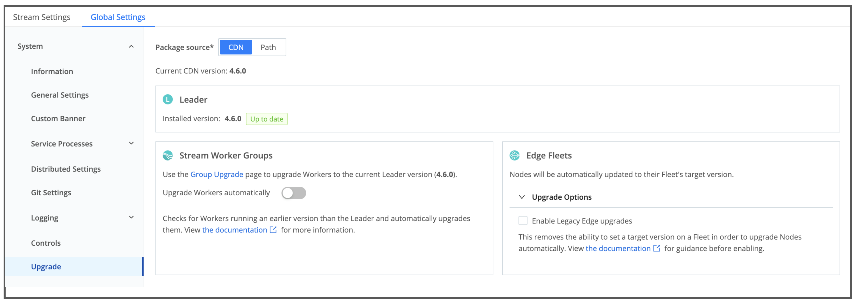
Selecting a Package Source

Here, you'll select between **CDN** and **Path** for your upgrade (in an on-prem, distributed deployment):

Package Source	Description	When You Select This Package Source
CDN	Downloads an installation package directly from Cribl's content delivery network.	With **CDN** selected, you can see the currently installed version and the version available on the CDN. The Leader will always upgrade to the current CDN version when you click **Upgrade**.
Path	You designate the path to the installation package in the **Path** field.	You must designate the correct paths for your Leader and Worker Groups, both version and architecture. For example, a Worker Group that contains ARM64 Workers will require an ARM64 installation package.

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

Configuring Stream Settings for CDN Upgrade



Package source* CDN Path

Current CDN version: **4.6.0**

Leader

Installed version: **4.6.0** Up to date

Stream Worker Groups

Use the Group Upgrade page to upgrade Workers to the current Leader version (**4.6.0**).

Upgrade Workers automatically

Checks for Workers running an earlier version than the Leader and automatically upgrades them. View the documentation for more information.

Edge Fleets

Nodes will be automatically updated to their Fleet's target version.

Upgrade Options

Enable Legacy Edge upgrades

This removes the ability to set a target version on a Fleet in order to upgrade Nodes automatically. View the documentation for guidance before enabling.

Upgrade Stream using CDN

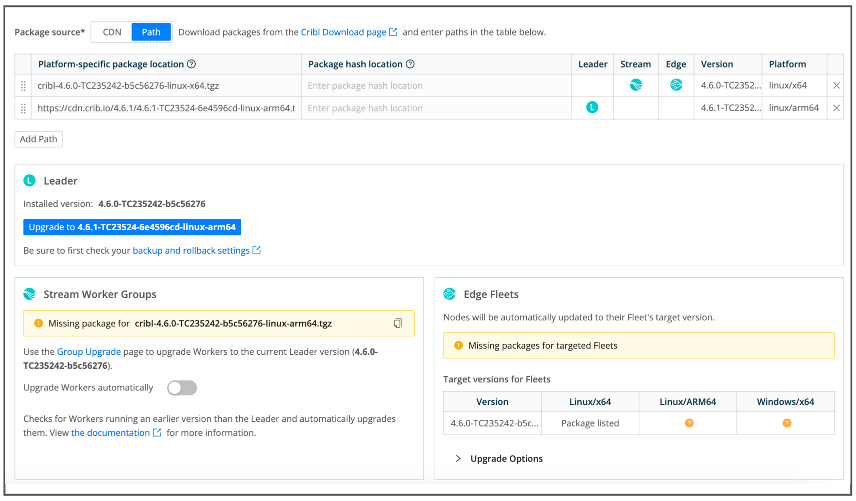
If you select **CDN** as your upgrade method, you will see the following options:

- **Current CDN version**: This lists the latest version of Cribl Stream available on the CDN.
- **Leader**: This shows the currently installed version of Cribl Stream on the Leader. If a newer version is available, you will be able to use the **Upgrade to** button.
- **Stream Worker Groups**: Provides a quick link to the Group Upgrade page if you want to upgrade Worker Groups individually.

  In Cribl.Cloud, this area will only show Hybrid Workers.

  To automatically upgrade Workers to the Leader version, enable **Upgrade Workers automatically**. See Automatic Upgrades for more details.
- **Edge Fleets**: Offers **Upgrade Options** for Edge. See Enable Legacy Edge Upgrades for more information.

Configuring Stream Settings for Path Upgrade



Package source* CDN Path Download packages from the Cribl Download page and enter paths in the table below.

Platform-specific package location	Package hash location	Leader	Stream	Edge	Version	Platform
cribl-4.6.0-TC235242-b5c56276-linux-x64.tgz	Enter package hash location		●	●	4.6.0-TC2352...	linux/x64
https://cdn.crib.io/4.6.1/4.6.1-TC23524-6e4596cd-linux-arm64.t	Enter package hash location	●			4.6.1-TC2352...	linux/arm64

Leader

Installed version: **4.6.0-TC235242-b5c56276**

Upgrade to **4.6.1-TC23524-6e4596cd-linux-arm64**

Be sure to first check your backup and rollback settings

Stream Worker Groups

Missing package for **cribl-4.6.0-TC235242-b5c56276-linux-arm64.tgz**

Use the Group Upgrade page to upgrade Workers to the current Leader version (**4.6.0-TC235242-b5c56276**).

Upgrade Workers automatically

Checks for Workers running an earlier version than the Leader and automatically upgrades them. View the documentation for more information.

Edge Fleets

Nodes will be automatically updated to their Fleet's target version.

Missing packages for targeted Fleets

Target versions for Fleets

Version	Linux/x64	Linux/ARM64	Windows/x64
4.6.0-TC235242-b5c...	Package listed	?	?

Upgrade Options

Upgrade Stream using Path

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

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

  Supports SHA-256 and MD5 formats. You can simply append `.sha256` to the contents of the **Platform-specific package location** field.

  When Stream is running in FIPS mode, MD5 is not an acceptable hashing algorithm.

- **Leader**: Indicates the version the Leader will be upgraded to when an upgrade is available and you click the **Upgrade to** button.
- **Stream**: The version in this row will be used to upgrade the Stream Worker Group, because it matches the Leader's version.
- **Edge**: The version in this row will be used to upgrade Edge Fleets with a matching target version.
- **Version**: Displays the package version in this row.
- **Platform**: Displays the platform architecture for the package in this row.

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

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

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

Stream Worker Groups: Provides a quick link to the Group Upgrade page if you want to upgrade Worker Groups individually.

To automatically upgrade Workers to the Leader version, enable **Upgrade Workers automatically**.

The **Edge Fleets** area offers **Upgrade Options** for Edge. See Enable Legacy Edge Upgrades and Configuring Edge Settings for Path Upgrade for more information.

Missing Package Links

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

To add package links:

1. Click the **Copy** icon to add the link to your clipboard.
2. Paste the copied package link into a new **Platform-specific package location** field.
3. Add the appropriate path directory to the beginning of the installation package path (where your Cribl upgrade packages are located).
   For example: `myfolder/directory/cribl-package.tgz`
4. Click **Save** to add the upgrade package path and resolve the warning.

Automatic Upgrades

You can enable **Upgrade Workers automatically** in **Global Settings** to check for Workers running an earlier version than the Leader, and automatically upgrade them to match the Leader.

Enable automatic upgrades has a different default state depending on the deployment type:

- In on-prem/customer-managed deployments, **Enable automatic upgrades** defaults to `No`, to prevent the Leader from automatically upgrading out-of-date Worker Nodes. Before you toggle this to `Yes`, upgrade the Leader itself to Cribl Stream's most recent version.
- In Cribl.Cloud deployments, **Enable automatic upgrades** defaults to `Yes`. This enables Cribl to automatically upgrade Worker Nodes to Cribl Stream's newest stable version. (Cribl-managed and hybrid Workers will auto-upgrade as soon as they see a new Leader version.) If you toggle this to `No`, you will need to explicitly upgrade each Worker.

When enabled, the automatic Worker upgrade process works like this:

1. The Leader pulls packages, and checks their hashes.
   - The Leader must be able to connect to the path.
   - The Leader must also have privileges to download the files.
   - If the path is an HTTP URL, the Leader copies the file to a known location in its filesystem.
   - If the package is already hosted on the Leader Node, specify its filesystem path.
2. Workers pull packages and check their hashes.
   - Workers pull from the Leader through HTTP, not directly from the Leader's filesystem.
   - Each Worker pulls the package that is appropriate for that Worker's platform and architecture.

3. Workers install the packages.

Upgrade Worker Groups via the UI

To upgrade a Worker Group:

1. Go to **Settings**, then **Stream Settings**.
2. Select **Group Upgrade**.
3. Click the **Upgrade** button in a row to upgrade that Group.

The resulting **Upgrade Group** modal offers two states: Basic Upgrade and Advanced Upgrade.

Upgrading Workers from the Leader requires a Cribl Stream Standard or Enterprise license.

Basic Upgrade Configuration

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

When one or more Workers can be upgraded in the Group, the **Upgrade** button will be enabled.

By clicking the **Upgrade** button, you can initiate the upgrade job with a task for each Worker that can be upgraded.

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

The **Worker Nodes** column will display a warning icon with a tooltip if the Group contains Workers that can't be upgraded for one or more of the following reasons:

- Workers are already running the current version.
- Workers running v.2.4.4 or older are too old to upgrade.

When none of the Workers in the Group can be upgraded, the **Upgrade** button will be disabled.

Advanced Upgrade Configuration

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

Quantity %: Specify what percentage of the Group's Workers to upgrade in this operation. If you enter a value less than the default `100%`, Cribl Stream will perform a partial upgrade, keeping the remaining

Workers active to process data.

Rolling upgrade: This option upgrades Workers in batches, each sized according to the value in the **Quantity %** field. When enabled, this toggle also enables the modal'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 confirm the Worker Group upgrade, the table on the Group Upgrade page will display an additional button on this Group'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, Cribl Stream first stores a backup of your current stable deployment. If the upgrade fails, then by default, Stream will automatically roll back to the stored backup package. You can adjust this behavior in **Settings**.

Go to **Global Settings**, then **System** and **General Settings**, then **Upgrade & Share Settings**.

Use the following controls to adjust backup and rollback behavior.

Cribl Stream can perform rollbacks only on Worker Nodes/instances that were running at least v.3.0.0 before the attempted upgrade.

Enable automatic rollback: Cribl Stream will automatically roll back an upgrade if the Cribl Stream 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 Rollback

Using CLI commands, it's possible to explicitly roll back an on-prem Leader, and on-prem or hybrid Workers, 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 Stream 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. Commit and push your Leader's configuration to that remote. The repo will provide a stable location from which to recover 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, Workers must not run a higher version than the Leader. Repeat steps 2, 3, and 5 above on all Workers (substituting "Worker" for "Leader").

Rollback Example

Although rollback can be partially scripted, you cannot discover earlier installation packages programmatically – the initial steps here require human intervention:

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 Workers to a compatible version.

Splunk App Package Upgrade Steps

See Deprecation note for v.2.1.

Follow these steps to upgrade from v.1.7, or higher, of the Cribl App for Splunk:

1. Stop Splunk.
2. Untar/unzip the new app version on top of the old one.
   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.
3. Restart Splunk.

Upgrading from Splunk App v.1.6 (or Lower)

As of v.1.7, contrary to prior versions, Cribl's Splunk App package defaults to Search Head Mode. If you have v.1.6 or earlier deployed as a Heavy Forwarder app, upgrading requires an extra step to restore this setting:

1. Stop Splunk.
2. Untar/unzip the new app version on top of the old one.
3. Convert to HF mode by running: `$SPLUNK_HOME/etc/apps/cribl/bin/cribld mode-hwf`

4. Restart Splunk.

9.9. Uninstalling

Removing an Cribl Stream installation, whether for a clean reinstall or permanently

Uninstalling the Standalone Version

- (Optional) To prevent systemd from trying to start Cribl Stream at boot time, run the following command:
  `sudo $CRIBL_HOME/bin/cribl boot-start disable`
  If you're running both Cribl Stream and Cribl Edge on the same host, be sure to execute this command in the same mode (Stream or Edge) and the same directory in which you originally ran `cribl boot-start enable`.
- Stop Cribl Stream (stopping the main process).
- Back up necessary configurations/data.
- Delete the `cribl` user: `userdel cribl`
- Delete the user group: `groupdel cribl`
- Remove the directory where Cribl Stream is installed.
- Directly delete any local files that Cribl Stream added while running – e.g., the `.dat` file and other local configs.

In a distributed deployment, repeat the above steps for the Leader instance and all Worker instances.

Uninstalling the Splunk App Version

- Stop Splunk.
- Back up necessary configurations/data.
- Remove the Cribl App in `$SPLUNK_HOME/etc/apps`.
- Remove the Cribl module in `$SPLUNK_HOME/etc/modules/cribl` (some versions).

10. Access Management

Cribl Stream provides a range of access-management features for users with different security requirements.

Where Can I Find Access Control Details?

See the following topics, according to your needs:

- Authentication: Authenticating users via local basic auth or external options (SSO, Splunk, LDAP).
- Members and Permissions: Available in Cribl Stream 4.2 and later. Fine-grained access control configurable at separate levels (Organization, product, Worker Group, and lower-level resources like Stream Projects and Search datasets).
- Local Users: Cribl Stream's original Role-based model for creating users, and for managing their access across a Cribl deployment.
- Roles: Cribl Stream's original RBAC model for managing Roles and Policies, and for assigning them to users.

Prerequisites (Restrictions on Restrictions)

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), note that **all users will have full administrative privileges.**

Which Access Method Should I Use?

Cribl currently supports both the new Members/Permissions and the legacy Users/Roles models, and these models are cross-compatible for many use cases. However, certain purposes require you to choose a specific model:

- **Cribl.Cloud** now relies only on Members/Permissions. See Cribl.Cloud Organization-level Permissions starting at Inviting Members, and product- and lower-level Permissions starting at Product-Level Permissions.

  > Cribl.Cloud's Organization-level Permissions include an Owner superuser. This option currently has no counterpart at the on-prem (customer-managed) Organization level.

- **Stream Projects** and Subscriptions, in Cribl Stream 4.2 and later, rely only on Members/Permissions. See Project-Level Permissions.
- **GitOps** integration authorization requires the legacy `gitops` Role. This legacy Role currently has no counterpart Permission.
- **Collectors**: The `collect_all` Role specifically enables creating, configuring, and running Collection jobs on all Stream Worker Groups. This legacy Role currently has no counterpart Permission.
- **Notifications**: The `notification_admin` Role specifically enables creating and receiving all Notifications. This legacy Role currently has no counterpart Permission.
- **Sources**, **Destinations**, **Pipelines**, and **Routes** are examples of other lower-level resources (below the product level) that can be shared with Local Users only by configuring custom access in legacy `policies.yml` configuration files.

  > Customizing these files is currently supported only with on-prem (customer-managed) deployments, not on Cribl.Cloud.

- **Search granular resources** (datasets, dataset providers, and search results) can be shared via Members/Permissions. For details, see the Search Sharing topic.

10.1. Authentication

User authentication in Cribl Stream

Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Worker Node, you set and push a new password to the Worker Group. Here's how, in the Leader Node's UI:

1. From the top nav, select **Manage**.
2. Select the desired Worker Group.
3. From the Worker Group'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 Stream when their assigned Roles change. Defaults to `Yes`.
- **Auth-token TTL**: Sets authentication tokens' valid lifetime, in seconds. Defaults to `3600` (60 minutes = 1 hour); must be at least `1` sec.
- **Session idle time limit**: Sets how long (in seconds) Cribl Stream 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 Stream sets tokens' valid lifetime by applying the **Auth-token TTL** field's value:

- When a user logs in, Cribl Stream 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 Stream's UI in their browser, Cribl Stream continually renews the token, resetting the idle-session time limit back by the **Auth-token TTL** value.

The cribl.secret File

When Cribl Stream 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 Stream! 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 Stream Roles and role mapping are supported **only** with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream user access Roles, see External Groups and Roles. The controls here are:

- **Default Role**: Default Cribl Stream 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 Stream Role in the right drop-down list. Click **Add Mapping** to add more rows.

10.2. Members And Permissions

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

In Cribl Stream 4.2 and later, 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 the current release, 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 Stream with the above prerequisites, you will be granted the Organization-level `Admin` Permission. Using this Permission, you can then assign additional Permissions to yourself and other Members, as outlined below in Organizations and Assigning Group-/Fleet-Level Permissions.

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?:

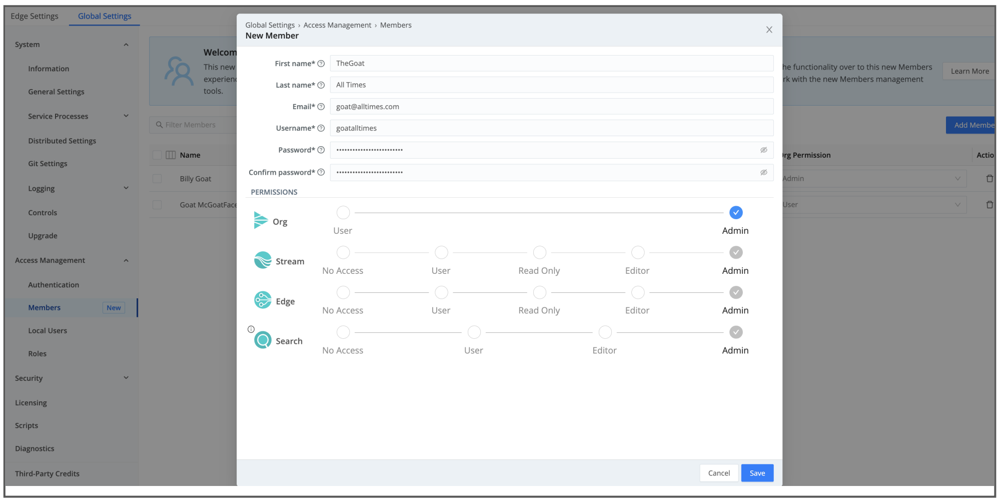
- Greater security in protecting access to resources and configuration.
- Greater flexibility in authorizing users on different parts of your Cribl suite (i.e., Organization).
- Ability to segment access and authorization to relevant teams.
- Ability to support a greater diversity of users, given these 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 Stream'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.


Global Settings › Access Management › Members
New Member

First name*: TheGoat
Last name*: All Times
Email*: goat@alltimes.com
Username*: goatalltimes
Password*: ························
Confirm password*: ························

PERMISSIONS

Org: User [✓ Admin]
Stream: No Access, User, Read Only, Editor, [✓ Admin]
Edge: No Access, User, Read Only, Editor, [✓ Admin]
Search: No Access, User, Editor, [✓ Admin]

Organization > Members/Permissions UI

For Cribl.Cloud's updated counterpart to this on-prem UI for managing Organization-level Permissions, see Inviting Members.

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 use the toggles to assign the following Permission levels.

For Cribl.Cloud Organization-level Permissions, see Cribl.Cloud docs' Member Permissions section.

Organization-Level Permissions

Permission	Description
User	The most basic Permission. At the Organization level, gets the Member into the system, without conferring any access to peer Members, products, or lower-level resources. (Admins can freely assign these Members varying Permissions at lower levels.)
Admin	This is a superuser Permission. At the Organization level, allows creating, viewing, updating, and deleting all Members. (Automatically inherits Product-level `Admin` Permissions and resource-level `Maintainer` Permissions, which confer comparably broad capabilities.)

For Stream and Edge, the Product-level Permissions below apply to both Cribl.Cloud and on-prem deployments. For Cribl Search's more-specific Product-level Permissions, see Search Member Permissions.

Product-Level Permissions (Stream and Edge)

Permission	Description
User	The most basic Permission. At the Product level, makes the Member assignable to Worker Groups 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, Worker Groups, 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 Worker Groups and lower-level resources.)

Permission	Description
Editor	Allows viewing all Groups and Monitoring pages. (Configuring these options requires `Admin`-level permission on the product and/or Worker Group.) Automatically inherits the `Editor` Permission on Worker Groups, 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 Worker Groups and resources; managing Worker Group Mappings; adding, updating, and restarting Worker 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 Worker Groups 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 Stream Permissions at the Worker Group level generally mirror those at the Product level:

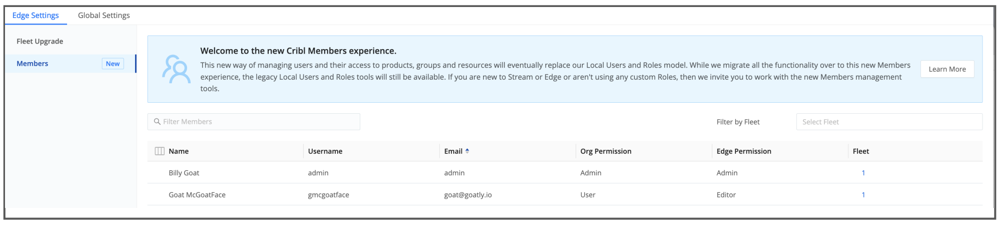
Permission	Description
User	The most basic Permission. Gets the Member into the Worker Group, where Admins can then assign them access on individual resources. (Unless Members have a higher Permission at the Product level, they have no initial access to Worker Groups or resources.)
Read Only	Designed for support personnel. Allows viewing all Worker Group-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 Worker Group-level encryption keys, certificates, secrets, scripts, Sources, Destinations, Pipelines, Packs, Routes, QuickConnect connections, Knowledge objects, and Notifications and Notification targets. Allows committing configuration changes, but not deploying them. (An `Editor` Permission at the Product level automatically inherits `Editor` Permissions on all Worker Groups.)
Admin	Superuser Permission. Allows creating, viewing, updating, and deleting all Worker Group-level access management (Members' Permissions), Settings, encryption keys, key management system (KMS) settings, certificates, secrets, scripts, Sources, Destinations,

Permission	Description
	Pipelines, Packs, Routes, QuickConnect connections, Knowledge objects, Notifications and Notification targets, and Stream Projects/Subscriptions. Allows adding, updating, and restarting Worker 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 Worker Group:

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 Worker Group.
3. From the top-left product switcher, select **Cribl Stream**. Then select (depending on the product) **Settings** > **Stream Settings** > **Members** or **Settings** > **Edge Settings** > **Members**.

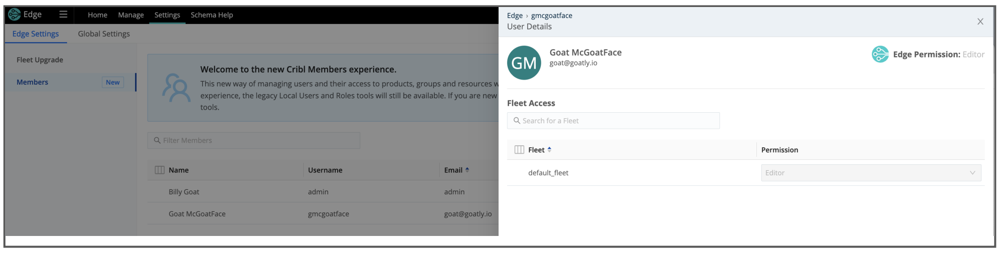


Welcome to the new Cribl Members experience.
This new way of managing users and their access to products, groups and resources will eventually replace our Local Users and Roles model. While we migrate all the functionality over to this new Members experience, the legacy Local Users and Roles tools will still be available. If you are new to Stream or Edge or aren't using any custom Roles, then we invite you to work with the new Members management tools.

Name	Username	Email	Org Permission	Edge Permission	Fleet
Billy Goat	admin	admin	Admin	Admin	1
Goat McGoatFace	gmcgoatface	goat@goatly.io	User	Editor	1

Edge Settings > Members

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



Goat McGoatFace
goat@goatly.io
Edge Permission: Editor

Fleet Access

Fleet	Permission
default_fleet	Editor

Product Members > User Details

5. Grant the Member the Permission level you want on this Worker Group.

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 Cribl Stream 4.2 and later, Stream Projects and Subscriptions rely entirely on the Members/Permissions access control first introduced in Stream 4.2. 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.

Other Resources

Access to certain resources can be managed only via legacy Local Users and Roles.

GitOps integration: Authorization to enable and manage GitOps requires the legacy `gitops` Role. This legacy Role currently has no counterpart Permission.

Collectors: The `collect_all` legacy Role specifically enables creating, configuring, and running Collection jobs on all Stream Worker Groups. This Role currently has no counterpart Permission.

Notifications: The `notification_admin` legacy Role specifically enables creating and receiving all Notifications. This legacy Role currently has no counterpart Permission.

Sources, Destinations, Pipelines, and Routes are examples of other lower-level resources that can be shared with Local Users only by configuring custom access in legacy `policies.yml` configuration files.

> Customizing these files is currently supported only with on-prem (customer-managed) deployments, not on Cribl.Cloud.

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 Worker Groups, and to the `Maintainer` Permission on lower-level resources.
- Members with the Product-level `Editor` Permission will be locked to the `Editor` Permission on Worker Groups, 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 Worker Groups and on lower-level resources.

UX Customization

Cribl Stream'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 Stream'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.

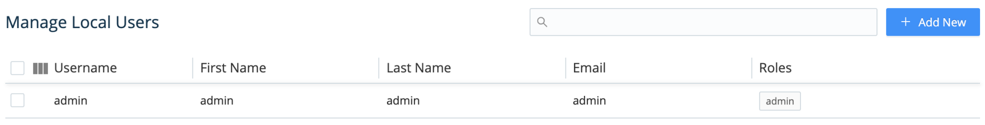
10.3. Local Users

This page covers how to create and manage Cribl Stream 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.



Manage Local Users

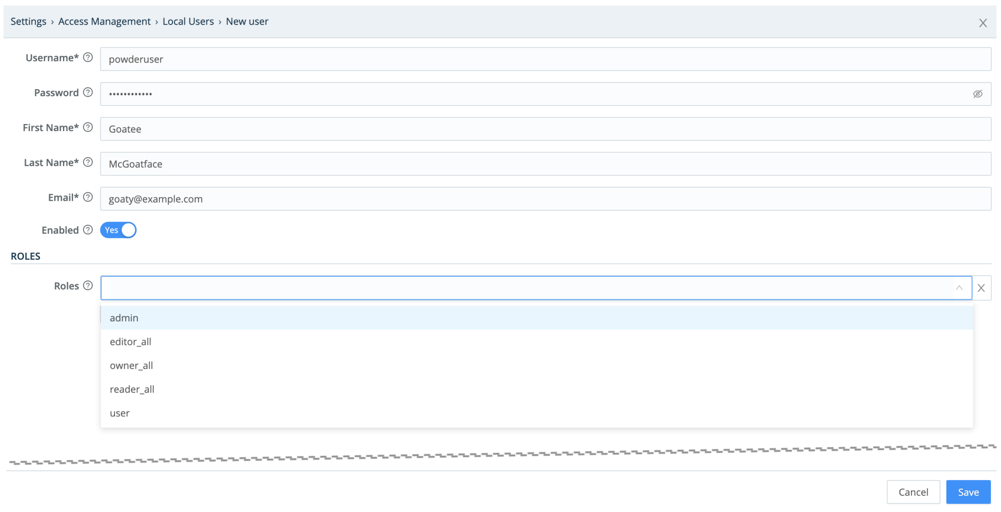
Username	First Name	Last Name	Email	Roles
admin	admin	admin	admin	admin

Managing users

To create a new Cribl Stream 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 Stream, but prevent them from currently logging in, you can toggle **Enabled** to `No`.



Settings › Access Management › Local Users › New user

Username*	powderuser
Password	•••••••••••
First Name*	Goatee
Last Name*	McGoatface
Email*	goaty@example.com
Enabled	Yes

ROLES

Roles: admin, editor_all, owner_all, reader_all, user

Entering and saving a user's credentials

Logged-in users can change their own Cribl Stream password from the **Local Users** page, by clicking on their own row to open a **Local Users** modal that manages their identity.

Password Rules

All passwords must:

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

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

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

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

Adding Roles

If you've enabled role-based access control you can use the modal's bottom **Roles** section to assign access Roles to this new or existing user.

> For details, see Roles. Role-based access control can be enabled only on distributed deployments (Edge, Stream) with 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 Stream 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.**

10.4. Roles

Define and manage access-control Roles and Policies

Cribl Stream 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 Stream 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 the current release, 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 Stream'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 Stream **users**.
- **Policies**: A set of **permissions**. A Role that is granted a given Policy can access, or perform an action on, a specified Cribl Stream object or objects.
- **Permissions**: Access rights to navigate to, view, change, or delete specified **objects** in Cribl Stream.
- **Users**: You map Roles to Cribl Stream users in the same way that you map **user groups** to users in LDAP and other common access-control frameworks.

> Users are independent Cribl Stream objects that you can configure even without RBAC enabled. For details, see Local Users.

How Cribl Stream RBAC Works

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

> In Cribl Stream 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 you cannot grant them finer-grained permissions on child objects such as Pipelines, Routes, etc. through the UI.
>
> Setting these granular permissions requires creating custom Policies in `policies.yml` configuration files. Note that this option is currently supported only with on-prem (customer-managed) deployments, not on Cribl.Cloud.

Cribl Stream'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 Stream'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 Stream 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 Stream ships with a set of default Roles, which you can supplement.

Default Roles

These Roles ship with Cribl Stream by default.

Organization-Level Roles

Note that some of the Roles below have no counterpart Permission in Cribl's newer Members/Permissions model.

Name	Description	Permission Equivalent
admin	Superuser – authorized to do anything in the system.	Organization Admin
gitops	Ability to sync the Cribl Stream deployment to a remote Git repository.	N/A
notification_admin	Read/write access to all Notifications.	N/A
user	Default Role that gets only a home/landing page to authenticate. This is a fallback for users who have not yet been assigned a higher Role by an admin.	Organization User
project_user	Read/write access to the simplified Stream Projects UI and related data Subscriptions. **Deprecated as of v.4.2.** Instead assign **Editor**, as the most comparable new Project-level Permission. The more-permissive **Maintainer**, and the more-restrictive **Read Only**, are also available Permissions.	Project Editor

Stream Roles

Name	Description	Permission Equivalent
stream_user	Basic Role for Stream.	Stream **User**.
stream_reader	Allows viewing all Members, Worker Groups, Settings, Leader commits, and legacy Local Users and Roles, with no configuration capabilities.	Stream **Read Only**.

Name	Description	Permission Equivalent
stream_editor	Allows viewing all Groups and Monitoring pages.	Stream Editor.
stream_admin	Superuser Role at the Product level	Stream Admin.

Edge Roles

Name	Description	Permission Equivalent
edge_user	Basic Role for Edge.	Edge User.
edge_reader	Allows viewing all Members, Fleets, Settings, Leader commits, and legacy Local Users and Roles, with no configuration capabilities.	Edge Read Only.
edge_editor	Allows viewing all Groups and Monitoring pages.	Edge Editor.
edge_admin	Superuser Role at the Product level.	Edge Admin.

Worker Group–Level Roles

Name	Description	Permission Equivalent
owner_all	Read/write access to (and Deploy permission on) all Worker Groups.	N/A
editor_all	Read/write access to all Worker Groups.	N/A
reader_all	Read-only access to all Worker Groups.	N/A
collect_all	Ability to create, configure, and run Collection jobs on all Worker Groups.	N/A

Search Roles

Name	Description	Permission Equivalent
search_user	Basic Role for Search.	Search User.
search_editor	Manage datasets, dataset providers, dashboards and settings.	Search Editor.
search_admin	Superuser Role at the Product level.	Search Admin.

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

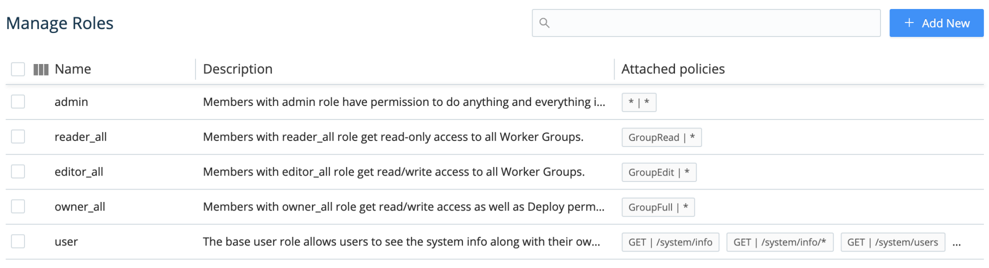
Initial Installation or Upgrade

When you first install Cribl Stream 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 Stream 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 Stream version's single Role.

Adding and Modifying Roles

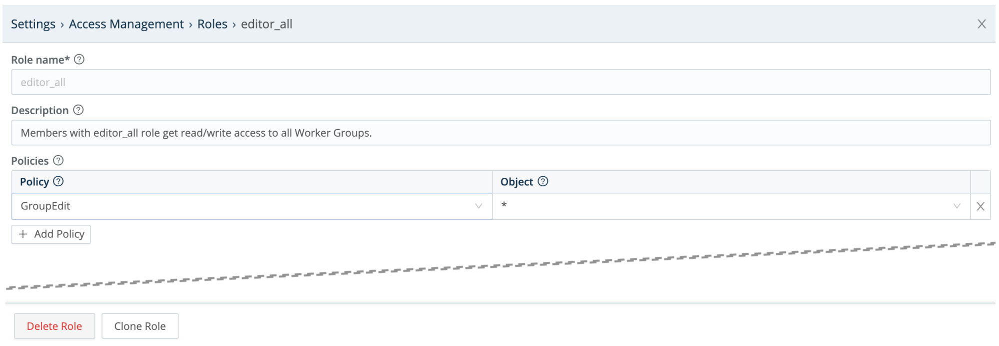
In a distributed environment, you manage Roles at the Leader level, for the entire deployment. On the Leader Node, select **Settings** > **Global Settings** > **Access Management** > **Roles**.



Name	Description	Attached policies
admin	Members with admin role have permission to do anything and everything i...	* | *
reader_all	Members with reader_all role get read-only access to all Worker Groups.	GroupRead | *
editor_all	Members with editor_all role get read/write access to all Worker Groups.	GroupEdit | *
owner_all	Members with owner_all role get read/write access as well as Deploy perm...	GroupFull | *
user	The base user role allows users to see the system info along with their ow...	GET | /system/info GET | /system/info/* GET | /system/users ...

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.


Settings › Access Management › Roles › editor_all

Role name*: editor_all

Description: Members with editor_all role get read/write access to all Worker Groups.

Policies

Policy	Object
GroupEdit	*

Add/edit Role modal

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

Role name: Unique name for this Role. Cannot contain spaces.

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.)

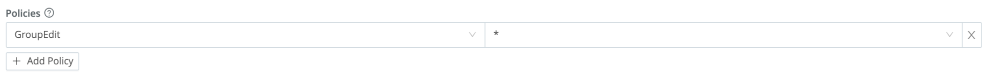
Clone Role: Opens a **New role** version of the modal, duplicating the **Description** and **Policies** of the Role you started with.

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 Streams 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

GroupEdit	*

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:

Worker Group-Level Policies

Name	Description	Permission Equivalent
`GroupRead`	The most basic Worker Group-level permission. Enables users to view a Worker Group and its configuration, but not modify or delete the config.	Worker Group-level Read Only.
`GroupEdit`	Building on `GroupRead`, grants the ability to also change and commit a Worker Group's configuration.	Worker Group-level Editor.
`GroupFull`	Building on `GroupEdit`, grants the ability to also deploy a Worker Group.	Worker Group-level Admin.
`GroupCollect`	Grants the ability to create, configure, and run Collectors on a Worker Group.	N/A
`GroupUser`	Access Worker Group.	Worker Group-level User.

Project-Level Policies

Name	Description	Permission Equivalent
`ProjectMaintain`	Grants ability to edit or delete the Project and its settings.	Project-level Maintainer.
`ProjectRead`	Can configure connections among the Project's Subscriptions, Packs, and Destinations, but cannot modify or delete these resources.	Project-level Read Only.

Name	Description	Permission Equivalent
`ProjectEdit`	Can view Project and Subscription settings and connections, but not modify or delete them.	Project-level Editor.

Product-Level Policies

Name	Description	Permission Equivalent
`ProductUser`	Makes the Member assignable to Worker Groups and resources, with no initial access to any.	Product-level User.
`LimitedProductUser`	Similar to `ProductUser`, but omits the ability to read or act on all the endpoints within a Worker Group.	N/A
`ProductAdmin`	Superuser Permission at the Product level.	Product-level Admin.

Search Policies

Name	Description	Permission Equivalent
`DatasetMaintain`	Full access to dataset configuration.	Search datasets Maintainer.
`DatasetRead`	Can view and search, but not modify dataset configurations.	Search datasets Read Only.
`DatasetProviderMaintain`	Full access to dataset provider configuration.	Search dataset providers Maintainer.
`DatasetProviderRead`	Can view and search, but not modify dataset provider configurations.	Search dataset providers Read Only.
`SearchUser`	Search data and view results shared with the user	Search Product-level User.
`SearchMaintainer`	Manage datasets, dataset providers, dashboards and settings.	Search Product-level Editor.
`DashboardMaintain`	Can edit and delete a Search dashboard.	Search dashboard

Name	Description	Permission Equivalent
		Maintainer.
`DashboardRead`	Can view and use a Search dashboard.	Search dashboard Read Only.

General Policies

Name	Description	Permission Equivalent
`*` (wildcard)	Grants **all** permissions on associated objects.	N/A

Internal Policies

The following policies are internal building blocks for Product-specific Policies. Do not add them directly to Roles.

Name	Description	Permission Equivalent
`Product`	N/A	N/A
`BaseProductUser`	N/A	N/A
`MaintainBase`	N/A	N/A
`SearchBase`	Similar to `SearchUser`: Can view Search resources shared with the Member.	N/A

Use Policies As-Is

By design, the default Policies that ship with Cribl Stream 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 Stream 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:

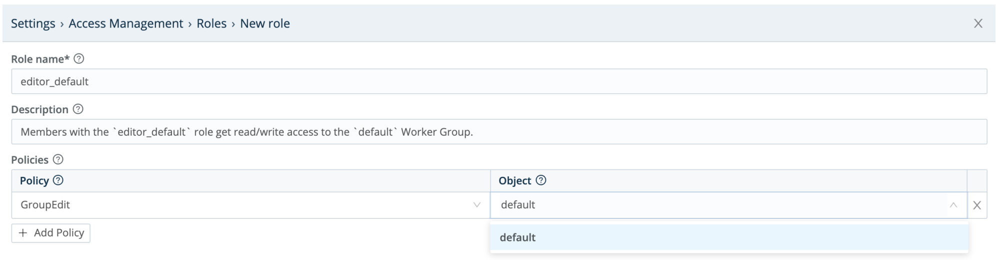
- `Worker Group <id>`
- `NewGroup2`
- `default` (Worker Group)
- `*` (all Worker Groups)

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 Stream.



Role name*: editor_default
Description: Members with the `editor_default` role get read/write access to the `default` Worker Group.
Policies

Policy	Object
GroupEdit	default

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 Stream 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.

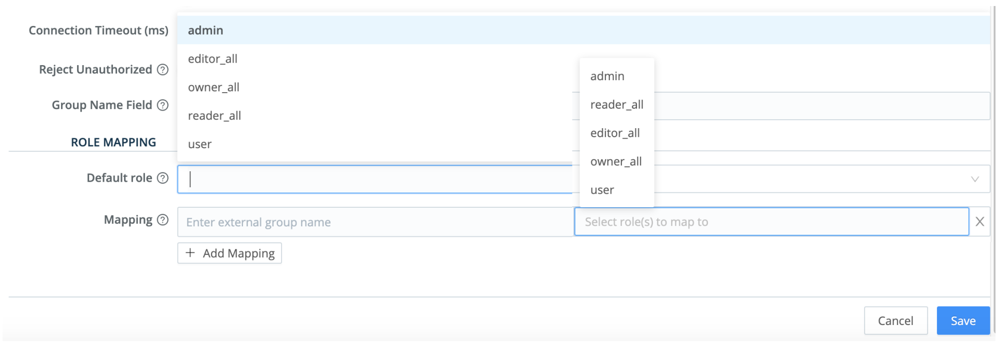
By default, Cribl Stream will log out a user upon a change in their assigned Roles. You can defeat this behavior at **Settings** > **Global Settings** > **General Settings > API Server Settings > Advanced > Logout on roles change.**

External Groups and Cribl Stream Roles

You can map user groups from external identity providers (LDAP, Splunk, or OIDC) to Cribl Stream Roles, as follows:

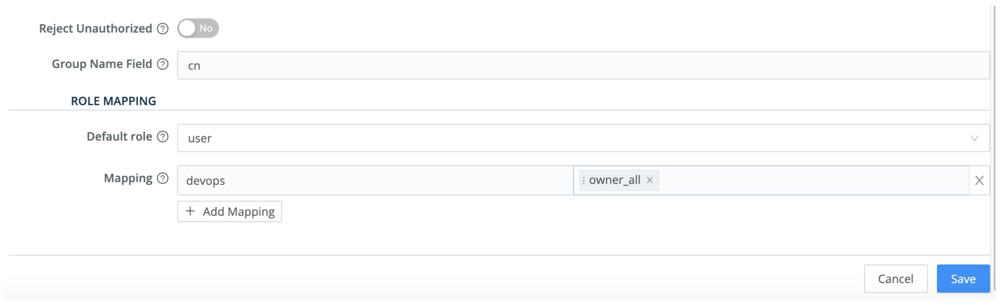
1. Select **Settings** > **Global Settings** > **Access Management** > **Authentication**.
2. From the **Type** drop-down, select **LDAP**, **Splunk**, or **OpenID Connect**, according to your needs.
3. On the resulting **Authentication Settings** page, configure your identity provider's connection and other basics. (For configuration details, see the appropriate Authentication section.)
4. Under **Role Mapping**, first select a Cribl Stream **Default role** to apply to external user groups that have no explicit Cribl Stream mapping defined below.
5. Next, map external groups as you've configured them in your external identity provider (left field below) to Cribl Stream 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 Stream Roles

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



Reject Unauthorized: No
Group Name Field: cn
ROLE MAPPING
Default role: user
Mapping: devops → owner_all

External user groups mapped to Cribl Stream Roles

10.5. SSO With Okta And OIDC

Cribl Stream 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 Stream 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 Stream 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 Stream instance.

The IDP can be Okta or Google, among others. Configuring SSO requires going back and forth between Cribl Stream 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 the @{product **Authentication** settings (with OIDC selected as the authentication method). 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 Stream! 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 Stream Roles

In Okta, admins organize their users in groups. In Cribl Stream, there are no user groups, but there are Roles. Your task includes **mapping** Okta groups to Cribl Stream Roles.

- Mapping groups to Roles is possible only for Cribl Stream deployments that are in Distributed mode, with an Enterprise license applied.
- If you are running Cribl Stream in Single-instance mode, you cannot map Okta groups to Cribl Stream Roles, although you can still set up SSO with Okta.

As you think through how best to map your Okta groups to Cribl Stream Roles, keep these principles in mind:

- An Okta group can map to more than one Cribl Stream Role.
- A Cribl Stream Role can map to more than one Okta group.
- If a user has multiple Roles, Cribl Stream applies the union of the most permissive levels of access.
- Cribl Stream 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 Stream Role(s)
a.	Cribl Admins	`admin`
b.	Cloud Admins	`reader_all`, `editor_cloud`
c.	Security Team	`reader_all`, `editor_cloud`, `editor_firewall`

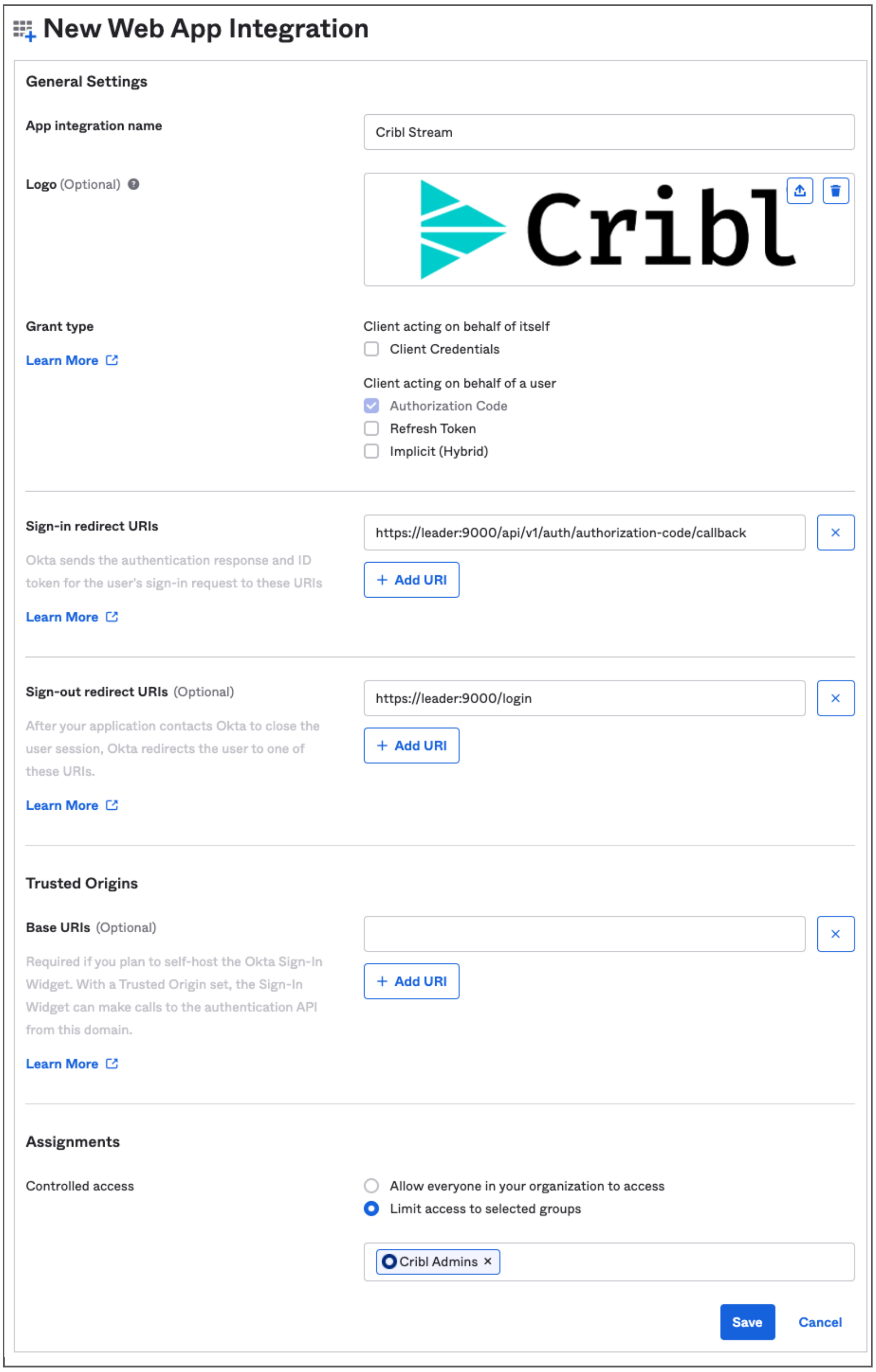
Cribl Stream Roles and role mapping are supported **only** with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Stream in the `admin` role.

Integrate Okta with Cribl Stream

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 Stream.
   - (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 Stream 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 Stream, select **Allow everyone in your organization to access**.
   - To permit specific Okta groups to access Cribl Stream, 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 ×

Completing the new app integration in Okta

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 Stream.

Begin Configuring OIDC Auth in Cribl Stream

> The only OAuth 2.0 flow that Cribl Stream supports is the Authorization Code Grant flow.
>
> In version 3.0 and higher, Cribl Stream's former "master" application components are renamed "leader." Above, while some legacy terminology remains within URLs, this document will reflect that.

In Cribl Stream, select **Settings** > **Access Management** > **Authentication**.

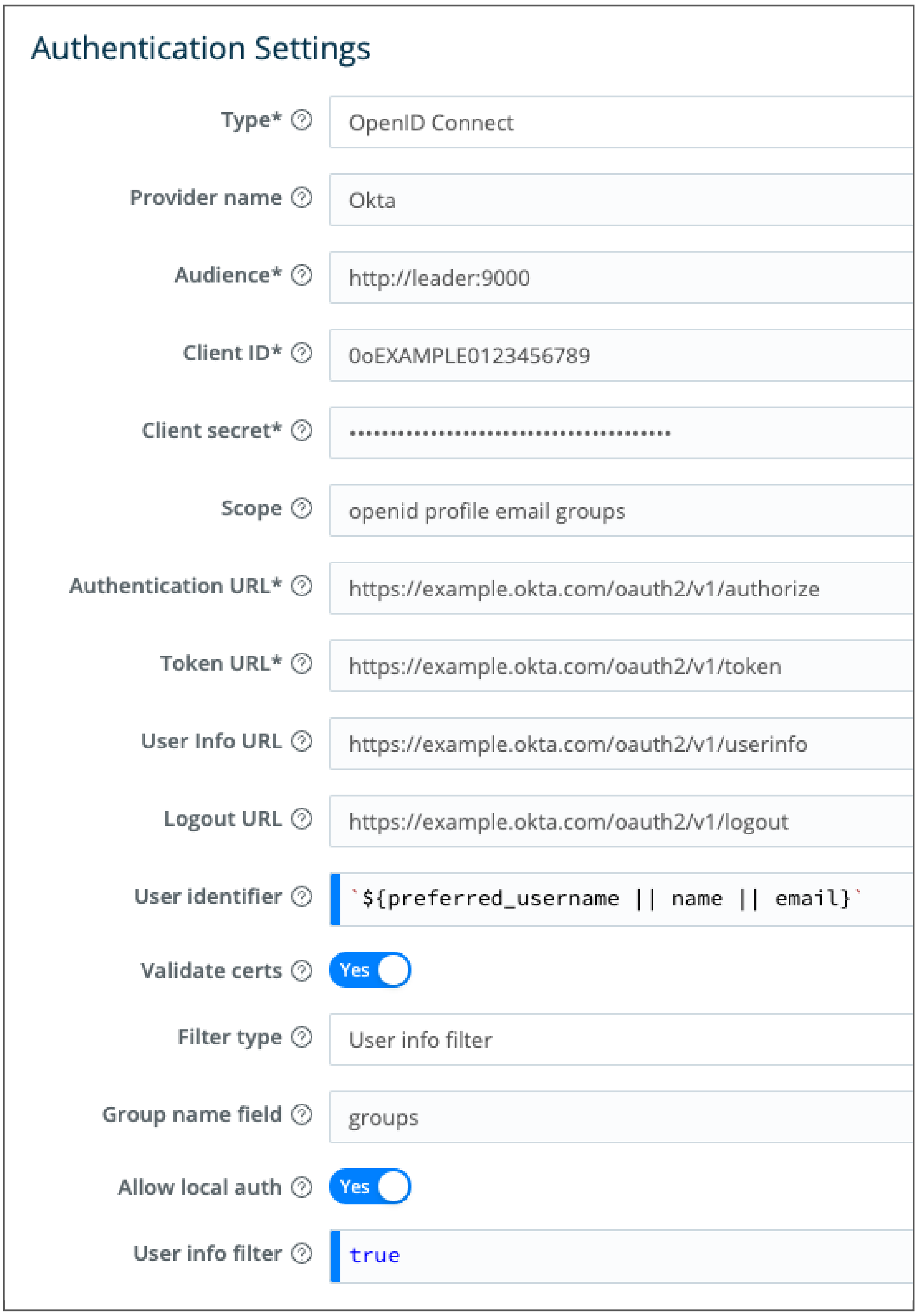
1. Choose `OpenID Connect` from the **Type** dropdown.
2. Choose `Okta` from the **Provider** dropdown.
3. In the **Audience** field, enter your Cribl Stream UI base URL. Do **not** append a trailing slash.
4. In the **Client ID** and **Client secret** fields, enter the respective values that you copied from the Okta UI in the previous step.
5. If your Cribl Stream 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 Stream.

9. If you want **Account** > **Log out** in Cribl Stream 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 Stream, they are logged out of **both** Cribl Stream and Okta.



Authentication Settings

Type*	OpenID Connect
Provider name	Okta
Audience*	http://leader:9000
Client ID*	0oEXAMPLE0123456789
Client secret*	••
Scope	openid profile email groups
Authentication URL*	https://example.okta.com/oauth2/v1/authorize
Token URL*	https://example.okta.com/oauth2/v1/token
User Info URL	https://example.okta.com/oauth2/v1/userinfo
Logout URL	https://example.okta.com/oauth2/v1/logout
User identifier	`` `${preferred_username || name || email}` ``
Validate certs	Yes
Filter type	User info filter
Group name field	groups
Allow local auth	Yes
User info filter	`true`

Authentication settings in Cribl Stream

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 Stream:

- For groups mastered inside Okta only, the app should pass a `Filter` type groups claim to Cribl Stream.
- For groups mastered outside Okta (such as Active Directory), or both inside and outside, the app should pass an `Expression` type groups claim back to Cribl Stream.

See the Okta documentation on dynamic allow lists and using Okta together with Active Directory.

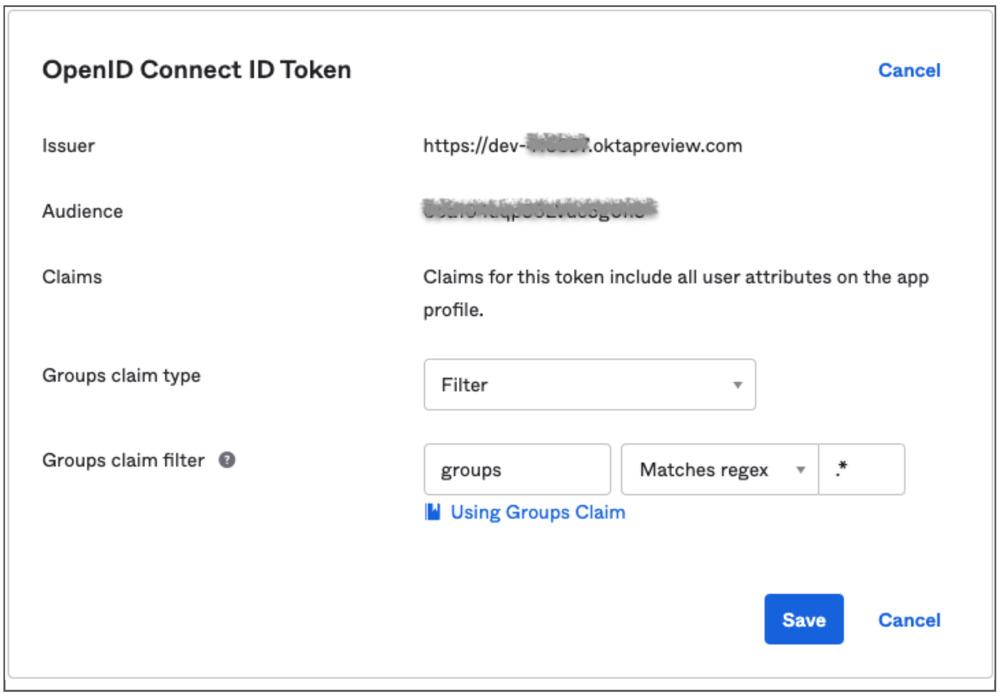
In Okta, you should still be in the panel for the app you created. If not, you can get there by opening **Applications** > **Applications** and selecting the app.

- 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**.



OpenID Connect ID Token

Issuer	https://dev-[redacted].oktapreview.com
Audience	[redacted]
Claims	Claims for this token include all user attributes on the app profile.
Groups claim type	Filter
Groups claim filter	groups / Matches regex / .*

Role mapping, beginning

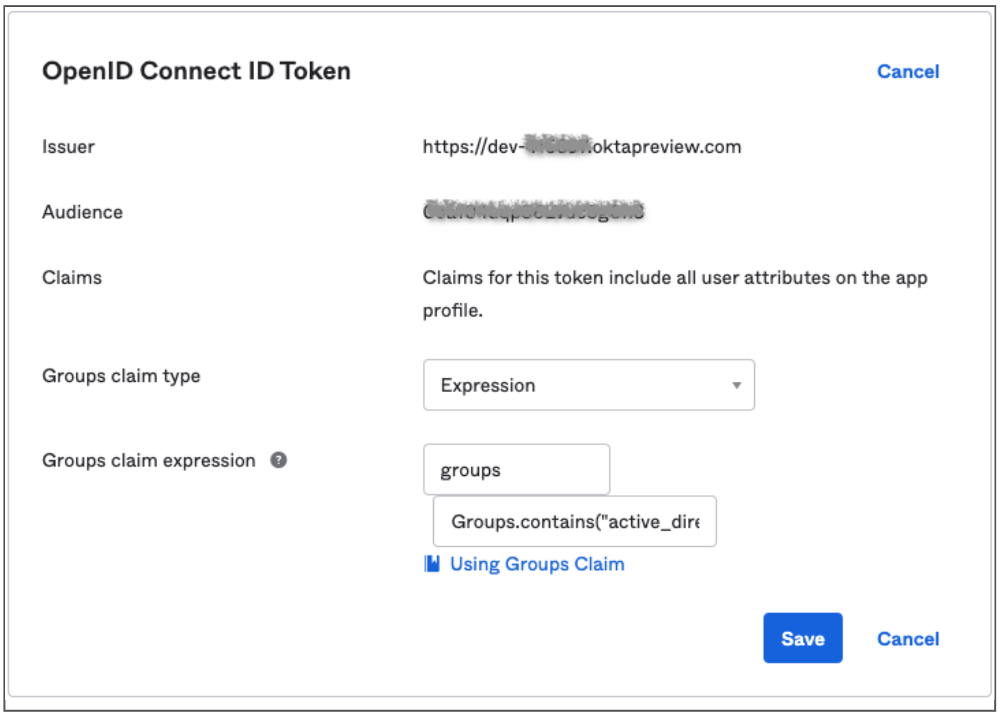
Configure Groups Outside of Okta

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 Stream. 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**.



OpenID Connect ID Token

Issuer	https://dev-[redacted].oktapreview.com
Audience	[redacted]
Claims	Claims for this token include all user attributes on the app profile.
Groups claim type	Expression
Groups claim expression	groups Groups.contains("active_dire

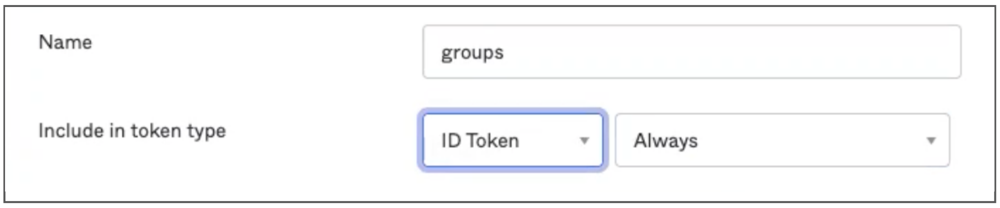
Role mapping, continued

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.



Name: groups

Include in token type: ID Token, Always

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 Stream

This section documents the entire Cribl Stream 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, for example: `https://leader.yourDomain.com:9000` for a distributed environment. For the IDP, this serves as a unique identifier for the SP (that is, your Cribl Stream 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`, for example: `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 Stream 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. For example: `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 Stream's local authentication. This enables an extra button called `Log in with local user` on the Cribl Stream 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 (for example, `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 Stream user access Roles, see External Groups and Roles. The controls here are:

- **Default role**: Default Cribl Stream 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).

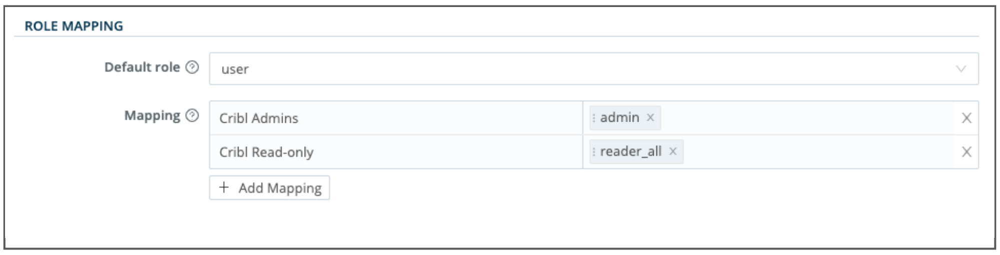
See also the explanation of role mapping above.

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 Stream Role to each Okta group name, and you can specify a `default` Role for users who are not in any groups.

1. In Cribl Stream, 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

Default role: user

Mapping	
Cribl Admins	admin
Cribl Read-only	reader_all

Role mapping, concluded

Verify that SSO with Okta Is Working

1. Log out of Cribl Stream, 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 Worker 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.

10.6. SSO With Okta And SAML

Cribl Stream 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 Stream 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 Stream 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 Stream instance.

The IDP can be Okta or Google, among others. Configuring SSO requires going back and forth between Cribl Stream 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 the Cribl Stream **Authentication** settings (with SAML 2.0 selected as the authentication method). 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 Stream! 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 Stream 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 Stream Roles

In Okta, admins organize their users in groups. In Cribl Stream, there are no user groups, but there are Roles. Your task includes **mapping** Okta groups to Cribl Stream Roles.

- Mapping groups to Roles is possible only for Cribl Stream deployments that are in Distributed mode, with an Enterprise license applied.

- If you are running Cribl Stream in Single-instance mode, you cannot map Okta groups to Cribl Stream Roles, although you can still set up SSO with Okta.

As you think through how best to map your Okta groups to Cribl Stream Roles, keep these principles in mind:

- An Okta group can map to more than one Cribl Stream Role.
- A Cribl Stream Role can map to more than one Okta group.
- If a user has multiple Roles, Cribl Stream applies the union of the most permissive levels of access.
- Cribl Stream 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 Stream Role(s)
a	Cribl Admins	`admin`
b	Cloud Admins	`reader_all`, `editor_cloud`
c	Security Team	`reader_all`, `editor_cloud`, `editor_firewall`

Cribl Stream Roles and role mapping are supported **only** with an Enterprise license. With a Standard license, all your external users will be imported to Cribl Stream in the `admin` role.

Begin Configuring SAML Auth in Cribl Stream

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 Stream instance, for example, `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 Stream 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 Stream

In this section, we'll create an Okta app that uses SAML and integrates with Cribl Stream.

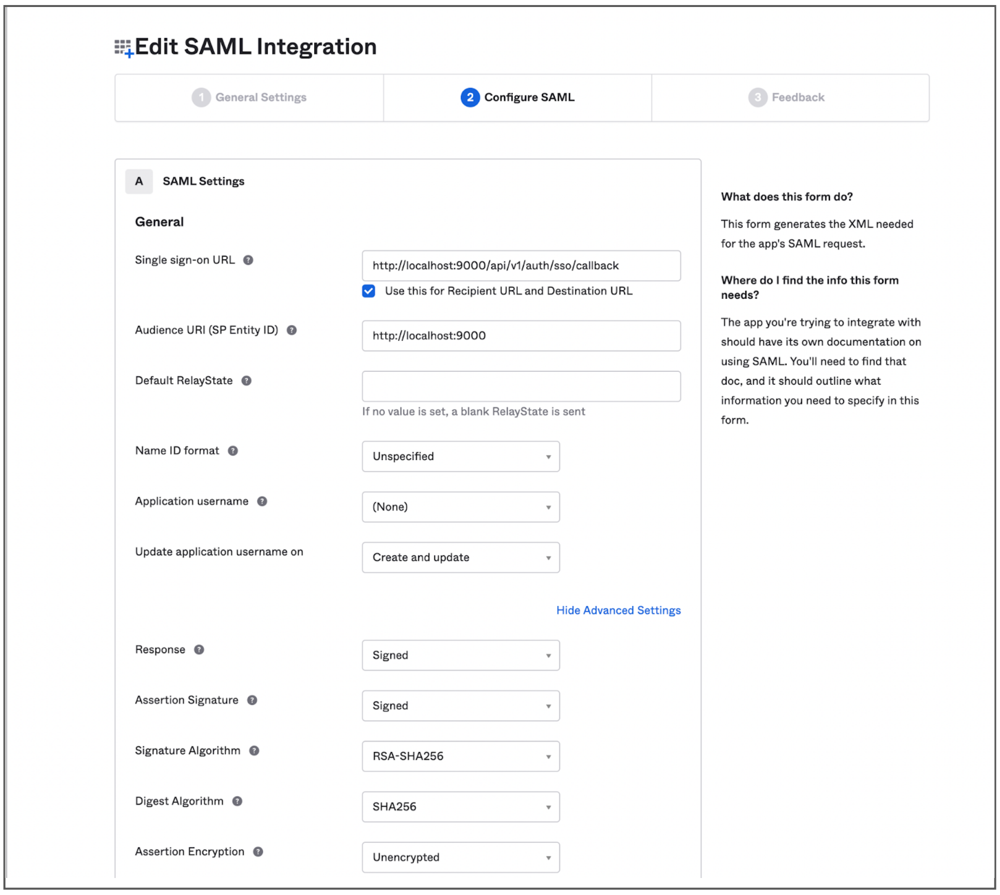
Configure General Settings (Part 1)

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 Stream.
   - (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 Stream **Sign-on callback URL**.
   - **Audience URI (SP Entity ID)**: Enter the Cribl Stream **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 Stream 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 Stream 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.


Edit SAML Integration

1 General Settings | 2 Configure SAML | 3 Feedback

A SAML Settings

General

Single sign-on URL	http://localhost:9000/api/v1/auth/sso/callback
	✓ Use this for Recipient URL and Destination URL
Audience URI (SP Entity ID)	http://localhost:9000
Default RelayState	If no value is set, a blank RelayState is sent
Name ID format	Unspecified
Application username	(None)
Update application username on	Create and update
	Hide Advanced Settings
Response	Signed
Assertion Signature	Signed
Signature Algorithm	RSA-SHA256
Digest Algorithm	SHA256
Assertion Encryption	Unencrypted

What does this form do?

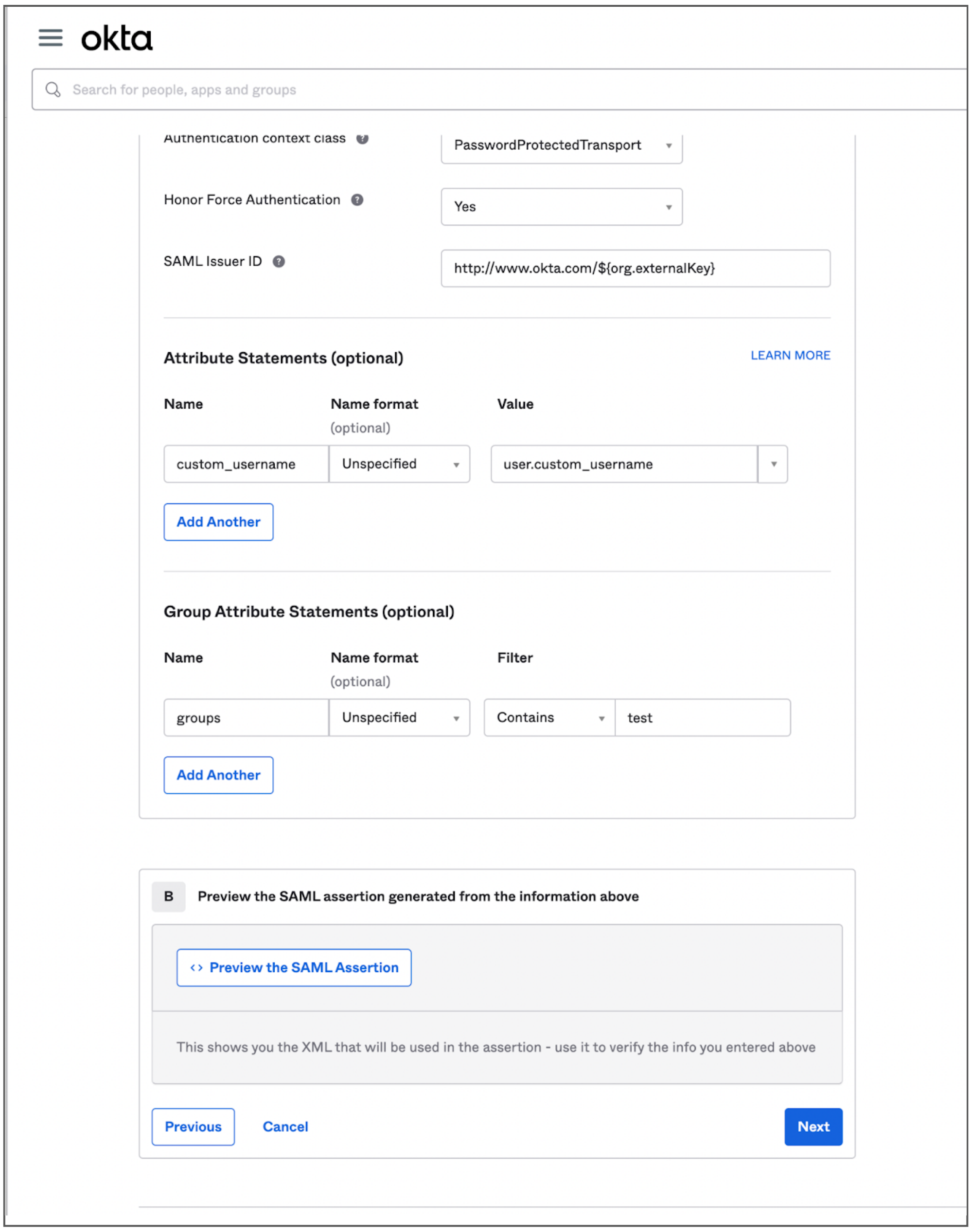
This form generates the XML needed for the app's SAML request.

Where do I find the info this form needs?

The app you're trying to integrate with should have its own documentation on using SAML. You'll need to find that doc, and it should outline what information you need to specify in this form.

Configuring SAML (Part 2)

2. (Optional) Define custom **Attribute Statements** that Okta will insert in the SAML Assertions shared with Cribl Stream. The only use case that Cribl Stream 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 Stream supports creating a custom attribute whose value is one or more Okta groups that will populate Cribl Stream'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**.


Authentication context class: PasswordProtectedTransport

Honor Force Authentication: Yes

SAML Issuer ID: http://www.okta.com/${org.externalKey}

Attribute Statements (optional)

Name	Name format (optional)	Value
custom_username	Unspecified	user.custom_username

Group Attribute Statements (optional)

Name	Name format (optional)	Filter	
groups	Unspecified	Contains	test

B Preview the SAML assertion generated from the information above

This shows you the XML that will be used in the assertion - use it to verify the info you entered above

okta
Search for people, apps and groups
LEARN MORE
Add Another
Add Another
Preview the SAML Assertion
Previous
Cancel
Next

Configuring SAML (Part 2) continued

Copy Your App's Metadata to Cribl Stream (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 Stream. Values that you did not configure above will not appear on the page.

SAML App Value	Cribl Stream Field
Identity Provider Single Sign-On URL	**Single sign-on (SSO) URL**

SAML App Value	Cribl Stream Field
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 Stream 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 Stream 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 Stream 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 Stream.

Digest Algorithm: Select your preferred hashing algorithm for the response that the IDP sends to Cribl Stream.

Assertion Encryption: Choose `Encrypted` if you want the IDP to encrypt the response that it sends to Cribl Stream. 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 Stream **Response decryption certificate** you selected or created in Cribl Stream.

Signature Certificate: Upload the public key of the Cribl Stream **Request certificate** you selected or created in Cribl Stream.

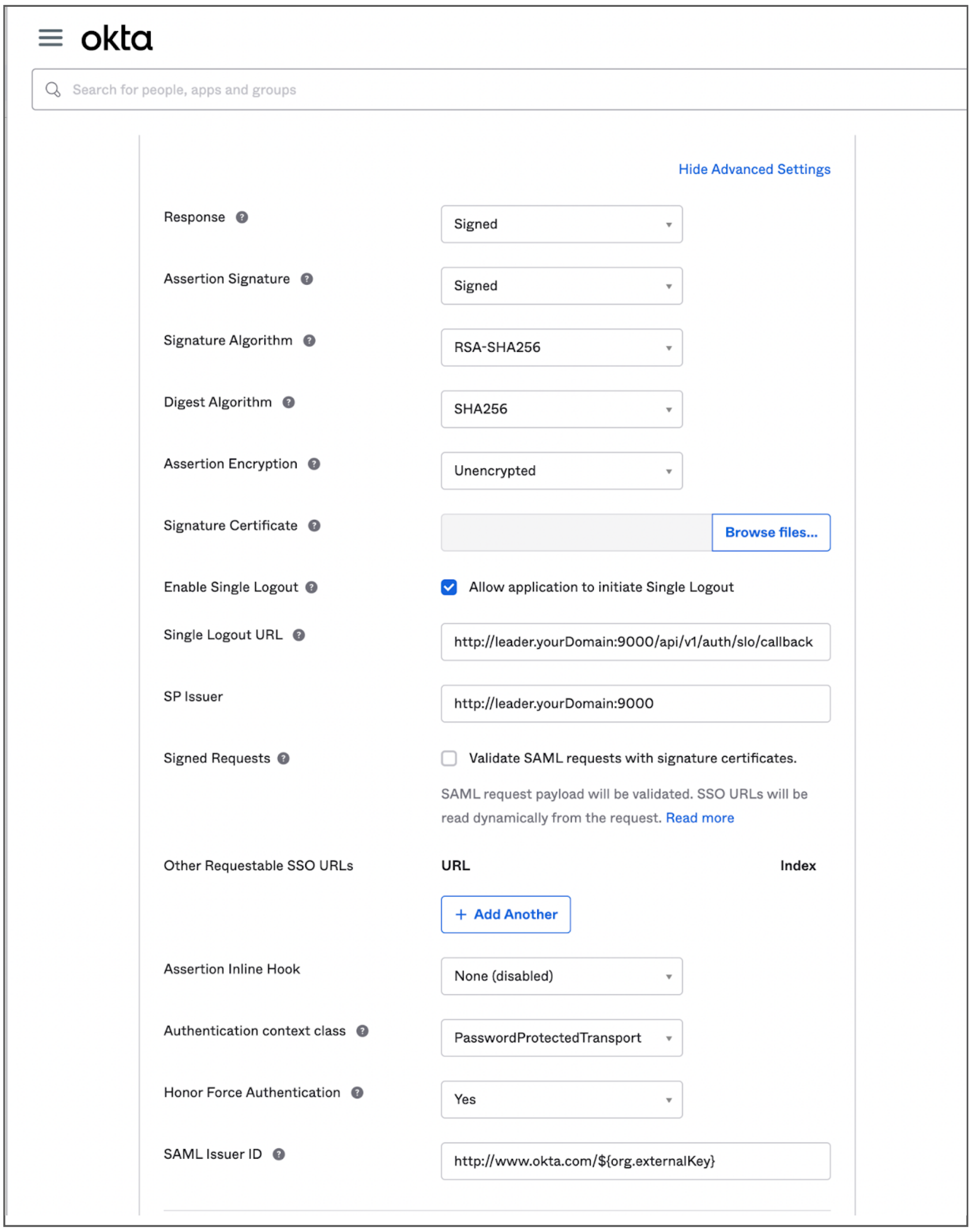
Enable Single Logout: Check the checkbox if you want Cribl Stream to send a logout request to Okta upon logout of the Cribl Stream user. Doing this displays the following two additional options that use values from earlier in these instructions:

- **Single Logout URL**: Enter the value that Cribl Stream provided for **Logout callback URL**.
- **SP Issuer**: Enter the URL that you entered for the **Audience** setting in the Cribl Stream UI.

Signed Requests: Check the checkbox if you want Okta to validate the signatures of SAML requests that come from Cribl Stream.

Other Requestable SSO URLs, **Assertion Inline Hook**, **Authentication context class**, and **Honor Force Authentication**: Ignore these settings (which are not supported by Cribl Stream).

SAML Issuer ID: Enables you to override Okta's default value for **Identity Provider Issuer**.



Response	Signed
Assertion Signature	Signed
Signature Algorithm	RSA-SHA256
Digest Algorithm	SHA256
Assertion Encryption	Unencrypted
Signature Certificate	
Enable Single Logout	✓ Allow application to initiate Single Logout
Single Logout URL	http://leader.yourDomain:9000/api/v1/auth/slo/callback
SP Issuer	http://leader.yourDomain:9000
Signed Requests	Validate SAML requests with signature certificates. SAML request payload will be validated. SSO URLs will be read dynamically from the request. Read more
Other Requestable SSO URLs	URL Index
Assertion Inline Hook	None (disabled)
Authentication context class	PasswordProtectedTransport
Honor Force Authentication	Yes
SAML Issuer ID	http://www.okta.com/${org.externalKey}

Okta Advanced Settings

Finish Configuring SAML Auth in Cribl Stream

This section documents the entire Cribl Stream 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, for example: `https://leader.yourDomain.com:9000` for a distributed environment. For the IDP, this serves as a unique identifier for the SP (that is, your Cribl Stream 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 Stream 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 Stream) will consume assertions (that is, will receive SAML authentication responses from the IDP).
- **Logout callback URL**: URL where the SP (Cribl Stream) will receive SAML logout responses from the IDP.
- **Metadata URL**: URL that exposes a description of the SP (Cribl Stream), 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 Stream) 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 (that is, 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 (that is, IDP's single logout service). Setting this will enable single logout initiated from the SP (Cribl Stream).

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 Stream) 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 Stream 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 Stream) 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 Stream 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:unspecifiec
    <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/AssertionC
    </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="ur
        <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:att
        <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlr
            test-admin
        </saml2:AttributeValue>
        <saml2:AttributeValue xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlr
            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 Stream.

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 Stream user access Roles, see External Groups and Roles. See also the explanation of role mapping above. The controls here are:

Default role: Default Cribl Stream 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).

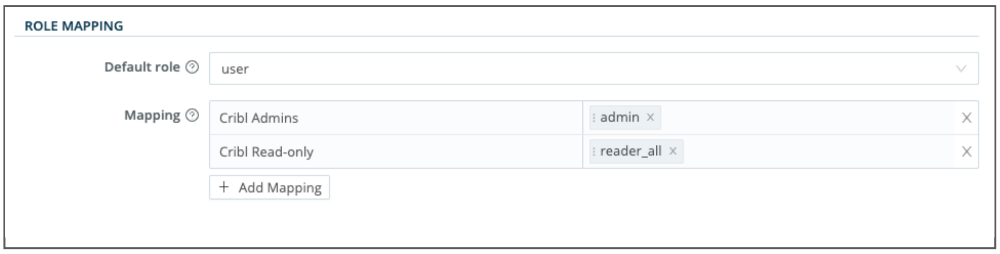
See also the explanation of role mapping above.

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 Stream Role to each Okta group name, and you can specify a `default` Role for users who are not in any groups.

1. In Cribl Stream, 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

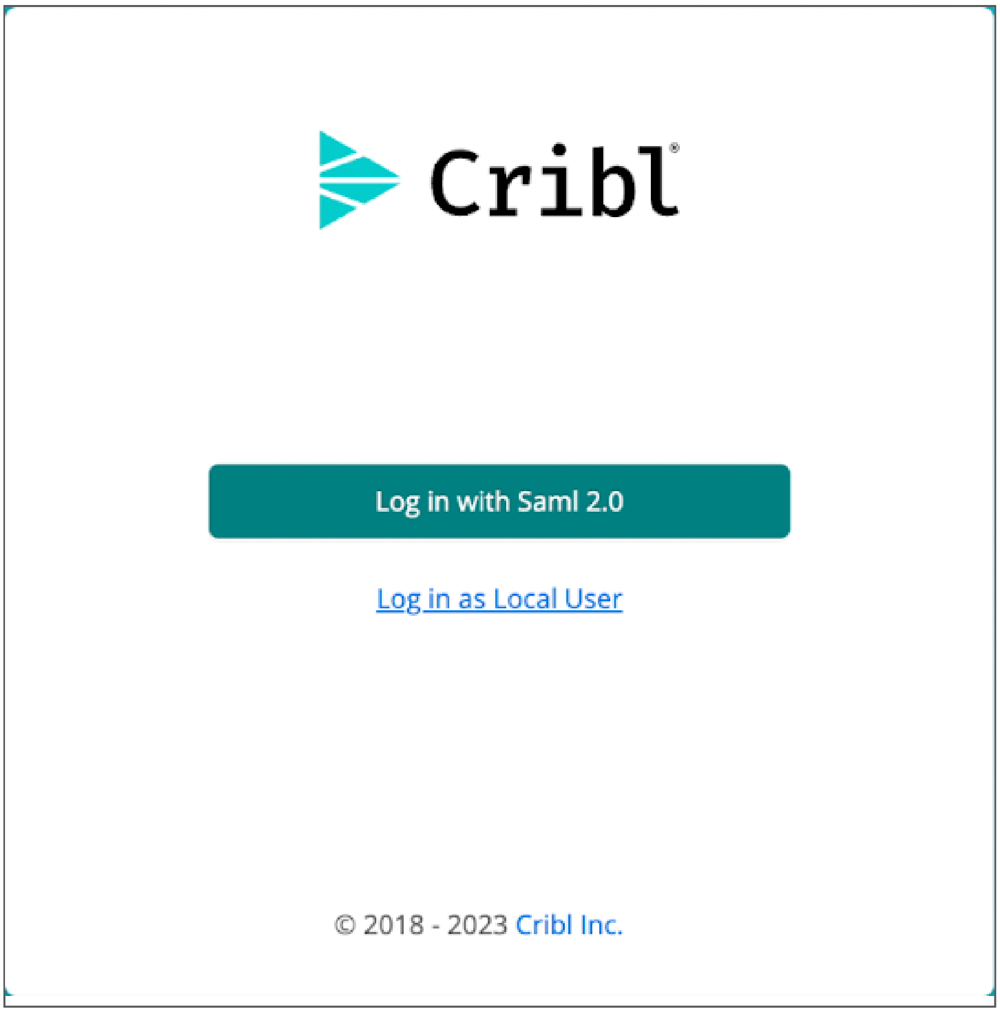
Default role: user

Mapping	
Cribl Admins	admin
Cribl Read-only	reader_all

Role mapping, concluded

Verify that SSO with SAML Is Working

1. Log out of Cribl Stream, 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 Worker 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 Stream will pick up because it uses the native AWS SDK.

You can set up multiple S3 Sources with different credentials. Cribl Stream 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.

11. Securing

11.1. FIPS Mode

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

Requirements

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

FIPS Mode System Requirements

To run Cribl Stream in FIPS mode, your environment must satisfy the following requirements:

- The operating system must support FIPS 140-2; several Linux distributions meet this standard. See NIST's list of tested configurations here.
- In particular, the environment must have a FIPS validated version of OpenSSL installed. For Cribl Stream version 4.5, this must be OpenSSL version 3.0.9 or 3.0.8, with its associated certificate.

FIPS Mode Restrictions on Cryptographic Algorithms

When run in FIPS mode, Cribl Stream uses only those cryptographic algorithms that satisfy the FIPS standards. This means that Cribl Stream does not run any part of its code that uses the MD5 (message-digest) or CRC-32 (Cyclic Redundancy Check 32) algorithms, or any other algorithm that does not support FIPS.

Therefore, in FIPS mode:

Cribl expressions **that rely on MD5 or CRC-32 will fail silently.** This in turn affects the behavior of parent Functions and their existing parent Pipelines.

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

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

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

FIPS Mode Password Rules

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

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

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

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

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

Running Cribl Stream in FIPS Mode

Once Cribl Stream has been started without FIPS mode enabled, you cannot put it into FIPS mode. You must enable FIPS mode as described in this section, after installing **but before starting** Cribl Stream.

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

Enabling FIPS Mode on a Non-systemd Distribution

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

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

     ```
     fips: true
     ```

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

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

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

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

   - Set `OPENSSL_MODULES` to the value of `MODULESDIR`.
   - Set `CRIBL_OPENSSL_DIR` to the value of `OPENSSLDIR`.
3. Start Cribl Stream.

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

Enabling FIPS Mode on a systemd Distribution

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

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

   ```
   openssl version -a
   ```

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

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

   For example, you might run this and get the output shown:

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

   This command creates `nodejs.cnf` – the configuration file that NodeJS uses to run in FIPS mode as Cribl Stream starts.
3. Run the following command to open an empty temporary file in an editor:

   ```
   systemctl edit cribl
   ```

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

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

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

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

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

When a `nodejs.cnf` File Already Exists

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

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

```
nodejs_conf = nodejs_init

.include /<placeholder>/fipsmodule.cnf

[nodejs_init]
providers = provider_sect

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

[default_sect]
activate = 1
```

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

Verifying that you are in FIPS mode

- On the Leader, search `cribl.log` for this message:

```
"level":"info","message":"running with FIPS enabled"
```

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

Log in as `admin` – you will be prompted to enter a FIPS compliant password.

11.2. Leader's Auth Token

For new installations, Cribl Stream generates a secure, random Leader's auth token. A new token is also created when you switch from Single-instance to the Distributed mode by configuring a node to be a Leader mode.

In versions before v4.5.0, the default token was `criblmaster`. It is not changed if you upgrade your deployment to v4.5.0 or later.

If you prefer to use your own token (or you are still using the older default of `criblmaster`), change the token value to a unique secure value.

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

Changing the Auth Token Value

Changing the auth token once you have Worker Nodes deployed will break communication between the nodes and the Leader.

We recommend changing the auth token *before* deploying Worker Nodes.

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.

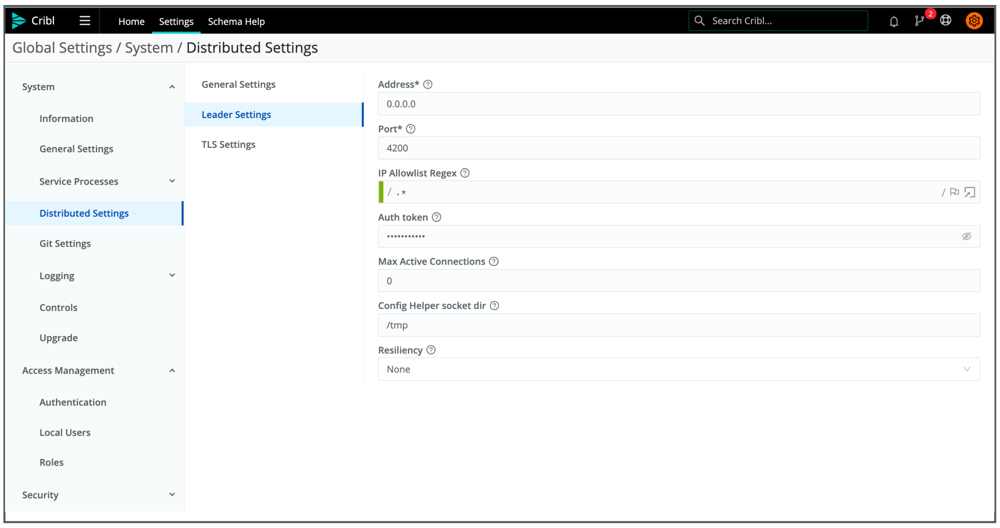
One option is to generate a token value with a shell command. For example, on Linux or Mac OS, the following command generates a 32-character value:

```
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, you can change it:

- in the **Auth token** field in the UI
- in the appropriate place in your Docker Compose file (if applicable)
- via command line: `./cribl mode-master -u <token>`

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



Address*: 0.0.0.0
Port*: 4200
IP Allowlist Regex: / .* /
Auth token: ••••••••••••
Max Active Connections: 0
Config Helper socket dir: /tmp
Resiliency: None

Distributed > Leader Settings

11.3. Securing Cribl Stream (TLS/SSL, Certs, Keys)

You can secure Cribl Stream 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. An additional option for Cribl Stream is to deploy in FIPS mode.

In a single-instance deployment (Stream, Edge), wherever this page refers to a Worker Group, just follow the Cribl Stream top nav's **Manage** link.

Secure Access to Worker Nodes' UI

A best practice in enterprise distributed deployments, this prevents direct browser access to Worker Nodes' UI.

Cribl recommends that you first enable the Leader's UI access to Worker Nodes (Stream, Edge). This way, admins will still be able to tunnel through from the Leader to any Worker Node's UI. This is also a prerequisite for Connecting Workers to the Leader Securely.

1. Select a Worker Group.
2. Open **Group Settings** (top right).
3. Under **General Settings** > **API Server Settings**, click **Advanced**.
4. Toggle **Local UI Access** to `No`.
5. Click **Save**.

SSL Certificate Configuration

You can secure Cribl Stream's API and UI access by configuring SSL. Do this on the Leader, to secure Worker 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 Stream expects certificates and keys to be formatted in privacy-enhanced mail (`.pem`) format.

In the Cribl Stream 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:

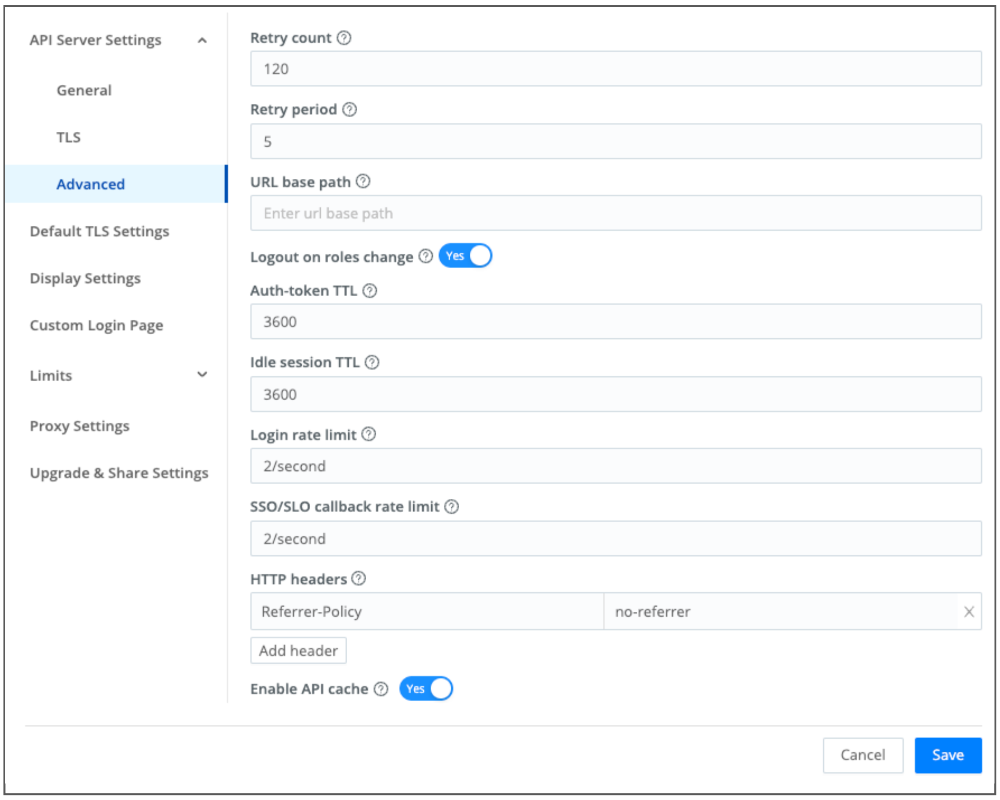
cribl.yml

```
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 Stream Leader and Worker 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.



Retry count: 120

Retry period: 5

URL base path: Enter url base path

Logout on roles change: Yes

Auth-token TTL: 3600

Idle session TTL: 3600

Login rate limit: 2/second

SSO/SLO callback rate limit: 2/second

HTTP headers: Referrer-Policy | no-referrer

Add header

Enable API cache: Yes

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 Stream	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 Stream (Source)	Default disabled	Default disabled	Default disabled
Data	Cribl Stream (Destination)	Any data receiver	Default disabled	Default disabled	Default disabled
Authentication	————	————	————	————	————
Local*	Browser	Cribl Stream	Default Disabled	N/A	N/A

Traffic Type	TLS Client	TLS Server	Encryption	Cert Auth	CN* Check
LDAP*	Cribl Stream	LDAP Provider	Custom	N/A	Default Disabled
Splunk*	Cribl Stream	Splunk Search Head	Default Enabled	N/A	Default Disabled
OIDC†/Okta*	Browser and Cribl Stream	Okta	Default Enabled	N/A	Enabled (Browser)
OIDC†/Google*	Browser and Cribl Stream	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 Stream'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`

TLS in Cribl.Cloud

TLS encryption is pre-enabled on several Sources in Cribl.Cloud, 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`

For steps to enable TLS mutual authentication in Cribl.Cloud, see Securing Communications.

Encryption Keys

You can create and manage keys that Cribl Stream 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 Stream 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 Worker Group, select **Settings** > **Security** > **Encryption Keys**.
- In a distributed deployment with multiple Worker Groups, keys are managed per Worker Group. 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 Stream will automatically generate this unique identifier.

Description: Optionally, enter a description summarizing this key's purpose.

Encryption algorithm: Currently, Cribl Stream supports the `aes-256-cbc` (default) and `aes-256-gcm` algorithms.

KMS for this key: Currently, the only option supported here is `local` (Cribl Stream'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.

Use initialization vector: If enabled, Cribl Stream will seed encryption with a nonce to make the key more random and unique. Optional (and defaults to disabled) with the `aes-256-cbc` algorithm; automatically enabled (and cannot be disabled) with the `aes-256-gcm` algorithm.

Initialization vector size: Length of the initialization vector (IV), in bytes. This option is displayed only with the `aes-256-gcm` algorithm. Defaults to `12` bytes to optimize interoperability, but you can use the drop-down to set this anywhere between `12–16` bytes.

Restoring Unencrypted Private Keys

If you need to roll back from Cribl Stream 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 Stream 4.1 (or later) on hosts – the Leader and all Worker Nodes.
2. Untar the older Cribl Stream version (e.g., 4.0.4) onto the Leader and all Worker Nodes.
3. Manually edit each Cribl Stream config file that contains an encrypted private key. Replace each key with its prior unencrypted version.
4. Start up Cribl Stream, commit and deploy, and resume normal operations.

Secrets

With Cribl Stream's secrets store, you can centrally manage secrets that Cribl Stream 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 Worker Group, select **Configure** > **Settings** > **Security** > **Secrets**.
- In a distributed deployment with multiple Worker Groups, secrets are managed on each Worker Group. 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 Stream'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 Stream 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 Stream trusts?** Set this environment variable in each Worker 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 Stream 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.

11.3.1. KMS Configuration

Cribl Stream's Key Management Service (KMS) maintains the keys that Cribl Stream uses to encrypt secrets on Worker Groups and Worker Nodes. The internal KMS is always available, but integrating an external KMS provider requires an Enterprise or Standard license.

To configure KMS for Cribl Stream 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 Worker Group are available at the Worker Group'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 Stream Settings' parallel Secrets section.
- HashiCorp Vault
- AWS KMS

External KMS Providers and Worker Groups

To integrate an external KMS provider into an on-prem distributed deployment, Cribl Stream'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 Worker Groups. 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 Worker Group.

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 Stream, 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 Stream Worker 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 Worker 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.

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for Vault Auth: Toggle to `Yes` if you want to use your Assume Role credentials to access Vault authentication.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming the role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

AWS EC2 Authentication

Vault Role: Enter the authentication role to use in Vault.

Custom auth path: If you enabled authentication in HashiCorp Vault with a custom path, enter that path again here. For example:

- You could have used the HashiCorp Vault command `vault auth enable -path /my-auth aws` to enable authentication with a custom path of `my-auth`. Subsequently, when you perform actions using the `vault write` command, you'd specify an auth type with the `auth_type=ec2` or `auth_type=iam` options.

Secret Engine

Mount: Mount point of the Vault secrets engine to use. (Currently, only the KVv2 engine is supported.) Defaults to `secret`.

Secret path: Enter the path on which the Cribl Stream secret should be stored, e.g.: `<somePath>/cribl-secret`.

> In a distributed deployment, the Leader, and each Worker Group, require a distinct secret. This location cannot be shared between them.

Advanced

Enable health check: Whether to perform a health check before migrating secrets data. Defaults to `Yes`.

Health check endpoint: Configurable endpoint to use for validating system health. Defaults to `/v1/sys/health`.

AWS KMS

Setting the **KMS provider** drop-down to `AWS KMS` exposes the following configuration options:

Authentication

Authentication method: Select an AWS authentication method.

- **Auto**: This default option uses the environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`, or the attached IAM role. Works only when running on AWS.
- **Manual**: You must select this option when not running on AWS.

The **Manual** option exposes these corresponding additional fields:

- **Access key**: Enter your AWS access key. If not present, will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM role credentials.
- **Secret key**: Enter your AWS secret key. If not present, will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for KMS: Toggle to `Yes` if you want to use Assume Role credentials to access the AWS KMS.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Service Configuration

KMS Key ARN: Enter the Amazon Resource Name (ARN) of the AWS KMS Key to use for encryption. This entry is required.

When you configure your IAM account/role in AWS, grant access to the following permissions on the KMS key that will be used:

- `kms:Encrypt`
- `kms:Decrypt`

Then use that account for authentication.

11.4. Securing Communications

This page outlines how to protect Cribl Stream Leader to Worker Nodes communications, using an existing TLS/SSL certificate and key.

Cribl Stream 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 Stream.

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.

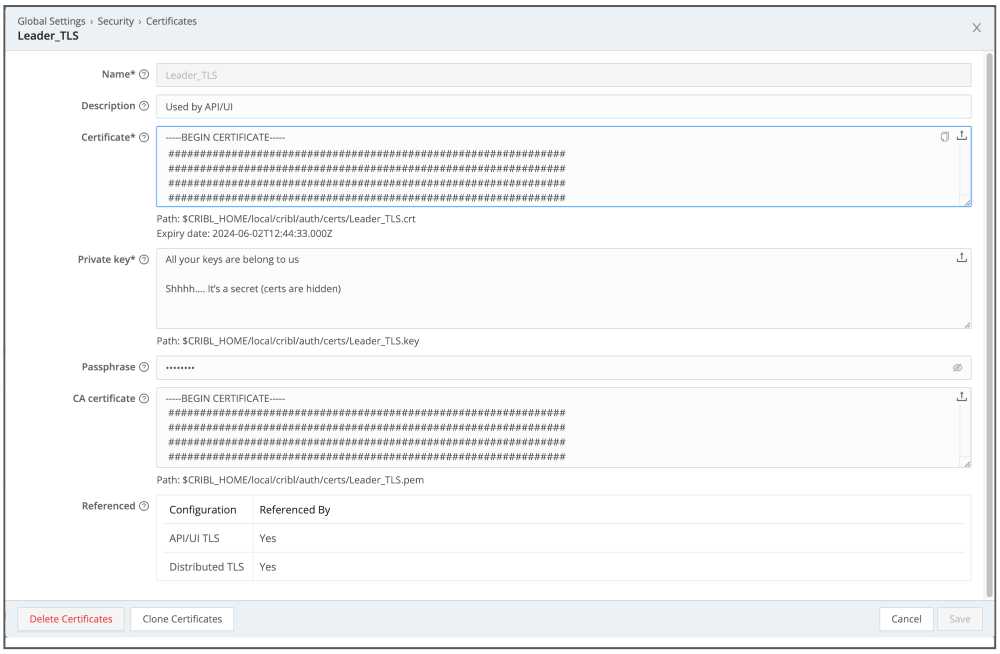
You can skip the preceding three steps: Just drag/drop your `.pem` file from your filesystem into the **Certificate** field, or upload it using the button at the field's upper right.

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).



Name*	Leader_TLS
Description	Used by API/UI
Certificate*	-----BEGIN CERTIFICATE----- ## ## ## ##
	Path: $CRIBL_HOME/local/cribl/auth/certs/Leader_TLS.crt Expiry date: 2024-06-02T12:44:33.000Z
Private key*	All your keys are belong to us Shhhh.... It's a secret (certs are hidden)
	Path: $CRIBL_HOME/local/cribl/auth/certs/Leader_TLS.key
Passphrase	••••••••
CA certificate	-----BEGIN CERTIFICATE----- ## ## ## ##
	Path: $CRIBL_HOME/local/cribl/auth/certs/Leader_TLS.pem

Referenced

Configuration	Referenced By
API/UI TLS	Yes
Distributed TLS	Yes

Leader's certificate modal, populated

Connecting to the Leader Securely

You can configure secure communication between your Leader and Worker Nodes using the UI, the `instance.yml` config file, or environment variables.

Using the UI

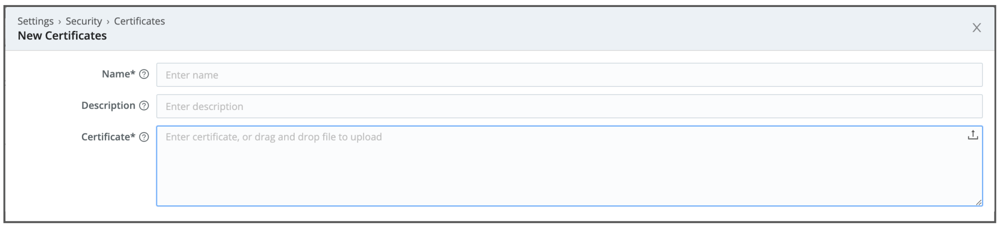
To set up secure communication via the UI, you configure first the Worker Group, then the Worker Node, then the Leader.

Worker Group Setup

For each Worker Group whose Worker 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 Worker Group 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 Worker Group's new configuration, including the new cert.
10. Repeat the preceding steps on each Worker Group.

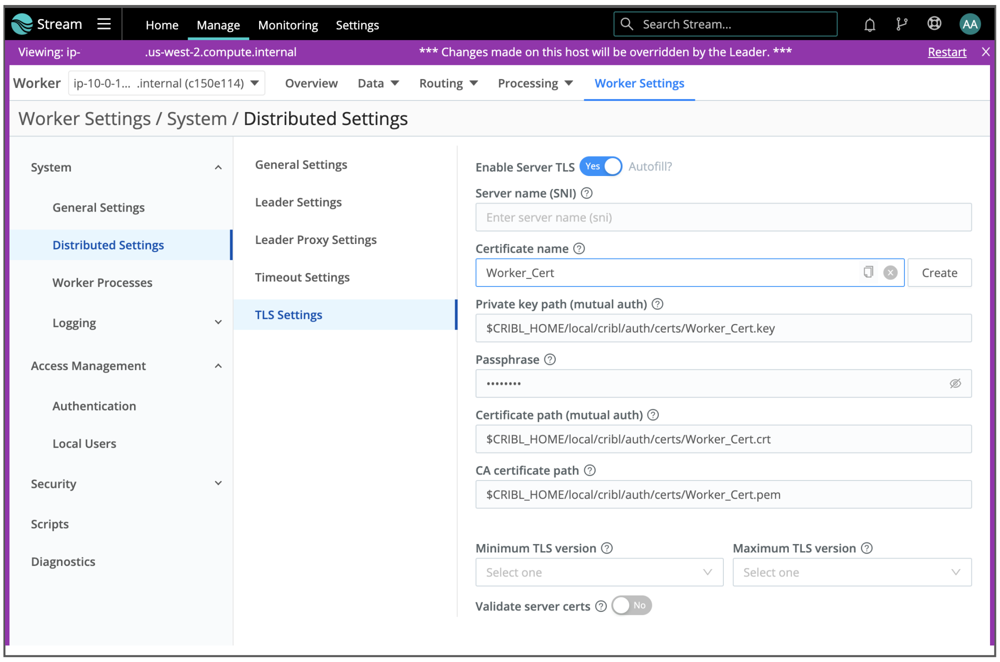

Settings › Security › Certificates
New Certificates
Name*
Enter name
Description
Enter description
Certificate*
Enter certificate, or drag and drop file to upload

Group-level certificates are configured like the Leader's cert

Worker Node Setup

For each Worker Node that you want to secure:

1. Enable the Leader's UI access to each desired Worker Node (Stream, Edge).
2. Tunnel through from the Leader to a Worker Node's UI.
3. Navigate to this Worker 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 Worker Group.
   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 Worker Node.



Enable Server TLS: Yes
Autofill?
Server name (SNI): Enter server name (sni)
Certificate name: Worker_Cert
Private key path (mutual auth): $CRIBL_HOME/local/cribl/auth/certs/Worker_Cert.key
Passphrase: ••••••••
Certificate path (mutual auth): $CRIBL_HOME/local/cribl/auth/certs/Worker_Cert.crt
CA certificate path: $CRIBL_HOME/local/cribl/auth/certs/Worker_Cert.pem
Minimum TLS version: Select one
Maximum TLS version: Select one
Validate server certs: No

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 Worker 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: <token>
    tls:
      disabled: false
      rejectUnauthorized: false
      requestCert: false
    resiliency: none
  group: default_fleet
```

After you've enabled TLS on the Leader, generating bootstrap scripts to add or update Worker Nodes will automatically prepend `https://` to the Leader's URL.

Using Environment Variables

Another way to set up secure communications between Worker Nodes and the Leader is via environment variables (Stream, Edge).

If you deploy your Worker 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 Stream 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 Stream. 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.

1. The Leader checks whether the `Common Name` (CN) matches the regular expression in the configuration. Cribl Stream'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

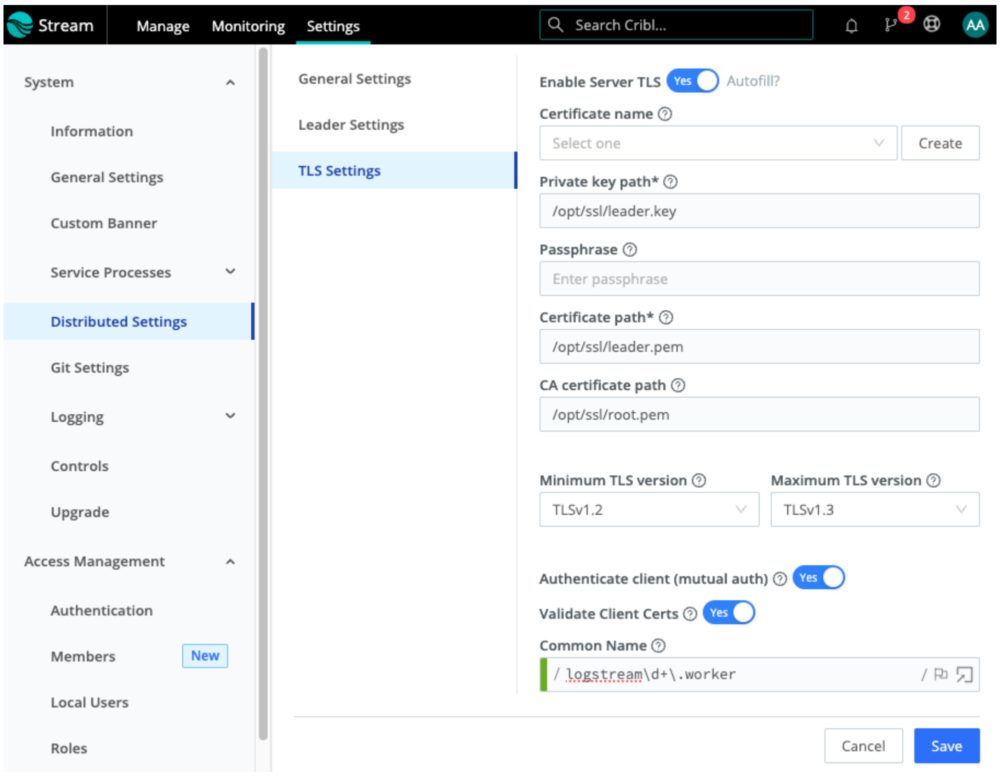
When configuring TLS mutual authentication on Worker Nodes, make sure you place your certificates into a separate directory outside of `$CRIBL_HOME`. If you place the certificates inside `$CRIBL_HOME`, they'll be removed when the next config bundle is deployed from the Leader.

Similarly, you can't bootstrap Worker Nodes with mutual authentication already populated. To bootstrap Worker 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

> TLS mutual authentication in Cribl.Cloud is configured separately for each Source.

Below is a sample configuration on the Leader:



Enable Server TLS: Yes — Autofill?

Certificate name: Select one

Private key path*: /opt/ssl/leader.key

Passphrase: Enter passphrase

Certificate path*: /opt/ssl/leader.pem

CA certificate path: /opt/ssl/root.pem

Minimum TLS version: TLSv1.2

Maximum TLS version: TLSv1.3

Authenticate client (mutual auth): Yes

Validate Client Certs: Yes

Common Name: `/ logstream\d+\.worker /`

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 Worker Node's `instance.yml` config file, you need to change some values and also add a few keys, as shown in this example:

```
distributed:
  mode: managed-edge
  master:
    host: <hostname>
    port: 4200
    authToken: <token>
    tls:
      disabled: false
      rejectUnauthorized: true # false if ignoring untrusted certs
      requestCert: true
      privKeyPath: /path/to/certs/worker.key
      certPath: /path/to/certs/worker.pem
      caPath: /path/to/certs/root.pem
    resiliency: none
  group: default_fleet
```

Using Environment Variables

You can set up 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/t
```

Once you've set this variable, restart the Worker Node. You should see the Worker Node successfully reconnect to the Leader.

If the Worker Node doesn't connect, check `cribl.log` on both the Worker Node and Leader for more context about the problem. You should see errors related to `dist leader communications`.

To build your own Certificate Authority (a self-signed CA), see our blog post on How to Secure Cribl Stream Worker-to-Leader Communications.

TLS Mutual Authentication on Cribl.Cloud

In Cribl.Cloud, you configure TLS mutual authentication separately for each Source.

This requires a CA certificate chain that can validate the client certificate used for authentication. You add your CA certificate by creating a new certificate entry in Cribl.Cloud.

1. Prepare the CA certificate chain PEM file.
2. Go to a Worker Node's **Settings** > **Security** > **Certificates** and select **Add Certificate**.
3. Populate the **Certificate** field with any valid PEM-formatted content:

   ```
   -----BEGIN CERTIFICATE-----
   CERTIFICATE CONTENT
   -----END CERTIFICATE-----
   ```

   The certificate and key are required only for UI validation and are not used otherwise.
4. Populate the **Private key** with the key in PEM format:

   ```
   -----BEGIN RSA PRIVATE KEY-----
   HIDDEN PRIVATE KEY
   -----END RSA PRIVATE KEY-----
   ```

5. In the **CA certificate** field, enter your PEM-formatted certificate and save. This will generate the CA certificate path.
6. Edit the certificate again to view the certificate path and copy it.
7. Go to the Source where you want to enable mutual TLS and paste the path in the **CA certificate path** field.
8. Save, commit, and deploy to finish the process.

Setting Authentication on Sources/Destinations

You can use certificates to authenticate Cribl Stream to external data senders and receivers. You configure this at the Group level, as follows:

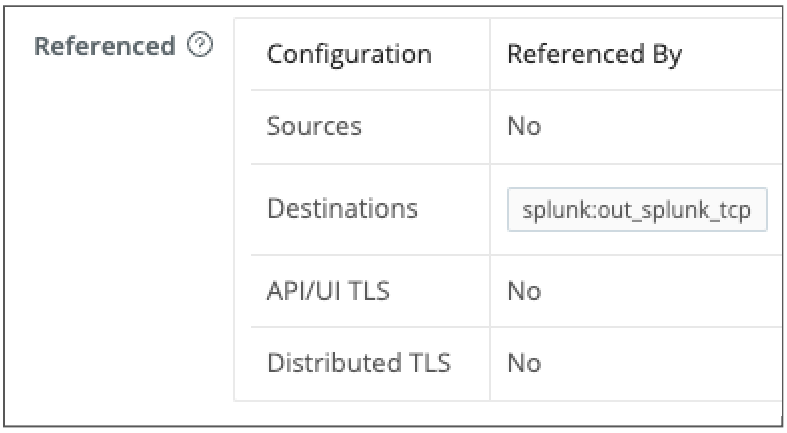
1. Select a Group.

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 Stream 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**.


Referenced

Configuration	Referenced By
Sources	No
Destinations	splunk:out_splunk_tcp
API/UI TLS	No
Distributed TLS	No

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 Stream:

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 Stream 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)`.

11.5. 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 Stream, 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. Some `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 an `HTTPS_PROXY` environment variable's referenced URL should generally be in `http://` format.

Preventing Proxy, Config, and Case Conflicts

Before you deploy Cribl Stream and configure proxies, Cribl recommends running the `printenv` shell command to check for any existing proxies in your environment. If you find existing proxies, be sure to include their environment variables in the systemd override file.

If Cribl Stream is already running on any Nodes you are proxying, restart the Cribl server after you initially configure – or make any changes to – the variables outlined on this page.

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 Stream endpoints rely on the HTTPS protocol. These include the Cribl telemetry endpoint (which must be accessed under some license types and all Cribl.Cloud plans), and the CDN used to propagate application updates and certain documentation features (API Reference and docs PDFs).

You might configure certain other Cribl Stream 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 Stream using systemd, add your proxy environment variables to the systemd override file (see Persisting Overrides). Add statements of this form:

Installed systemd File

```
[Service]
Environment=http_proxy=<yourproxy>
Environment=https_proxy=<yourproxy>
Environment=no_proxy=<no_proxy_list>
```

This will prevent Cribl Stream 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 username 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
```

If your `username` or `password` contains special characters, Cribl Stream will try to use these fields as the proxy address. As a workaround, URL-encode these fields.

Bypassing Proxies with `NO_PROXY`

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

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

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

`NO_PROXY` Usage

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

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

To match subdomains, you must either list them all in full (for example, `NO_PROXY=foo.example.com,bar.example.com`), or apply a wildcard by prefixing the domain name with a period or `*.`: `NO-PROXY=".example.com` or `NO-PROXY="*.example.com`.

To match the whole domain including its subdomains, add it both with and without wildcard to the list: `NO-PROXY="example.com,.example.com`.

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

Cloud `NO_PROXY` Usage

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

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

Where Proxies Apply

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

Destinations

- Amazon CloudWatch Logs
- Amazon Kinesis Streams
- Amazon S3 Compatible Stores

- Amazon SQS
- Azure Blob Storage
- Azure Monitor Logs
- Cribl HTTP
- CrowdStrike Falcon LogScale
- Datadog
- Data Lake > S3
- Elasticsearch
- Google Chronicle
- Google Pub/Sub
- Grafana Cloud
- Honeycomb
- Loki
- New Relic Ingest: Logs & Metrics
- New Relic Ingest: Events
- OpenTelemetry
- Prometheus
- SentinelOne DataSet
- SignalFX
- Splunk HEC
- Sumo Logic
- Wavefront
- Webhook

Sources

- Amazon SQS
- Amazon S3
- Azure Blob Storage
- CrowdStrike FDR
- Office 365 Services
- Office 365 Activity
- Office 365 Message Trace
- Prometheus Scraper

- Prometheus Edge Scraper

Collectors

- Amazon S3 Collector
- Azure Blob Storage Collector
- Google Cloud Storage Collector
- Health Check Collector
- REST/API Collector

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 Stream instance, and can validate that outgoing requests (to Destinations) are working properly.

Proxying Multiple Cribl Stream Instances in One Browser

Cribl Stream stores authentication tokens based on each http header's URI scheme, host, and port information. Within a given browser, Cribl Stream enforces a same-origin policy to isolate instances.

This means that if you want to run multiple proxied Cribl Stream instances in one browser session, you must assign them different URI schemes, hosts, and/or ports. Otherwise, logging into an extra Cribl Stream instance will expire the prior instance's session and log it out.

For example, assume that you've set up this pair of Apache proxy forward rules:

- https://web/cribla forwards to `cribl_hosta:8001/cribla`.
- https://web/criblb forwards to `cribl_hostb:8001/criblb`.

These two proxied addresses cannot be run simultaneously in the same browser session. However, this pair – which lead with separate URI schemes – could:

- https://web/cribla forwards to `cribl_hosta:8001/cribla`.
- https://web2/criblb forwards to `cribl_hostb:8001/criblb`.

Where separate instances **must** share URI formats, a workaround is to open the second instance in an incognito/private browsing window, or in a completely different browser.

11.6. Securing Data

Cribl Stream can be used to encrypt sensitive data in real time, and to route the encrypted data to an end system. Decrypted retrieval can be implemented on a per-system basis. Currently, decryption is supported only when Splunk is the end system.

- Data Encryption
- Data Decryption

11.6.1. Encryption

This page covers creating, modifying, and synchronizing symmetric keys.

Encryption of Data in Motion

Cribl Stream enables you to encrypt fields or patterns within events in real time, by using a `C.Crypto.encrypt()` expression in a Mask Function. The Mask Function accepts multiple replacement rules, and multiple fields to apply them to.

In that Function's configuration, a **Match Regex** defines the pattern of content to be replaced. The **Replace Expression** is a JavaScript expression or literal to replace matched content. You can use the `C.Crypto.encrypt()` method here to generate an encrypted string from a passed value.

Encryption Keys

You can configure symmetric keys through the CLI or UI. Users are free to define as many keys as required. Each key is characterized by the following:

- `keyId`: ID of the key.
- `algorithm`: Algorithm used with the key
- `keyclass`: Cribl Key Class (below) that the key belongs to.
- `kms`: Key management system for the key. Defaults to `local`.
- `created`: Time (epoch) when key was generated.
- `expires`: Time (epoch) after which the key is invalid. Useful for key rotation.
- `useIV`: Flag that indicates whether or not an initialization vector was used.

Key Classes

Key Classes in Cribl Stream are collections of keys that can be used to implement multiple levels of access control. Users (or groups of users) with access to data with encrypted patterns can be associated with key classes, for even more granular, pattern-level compartmentalized access.

Example

Users `U0, U1` have been given access to keyclass `0` which contains key IDs `0` and `1`. These keys are used to encrypt certain patterns in `datasetA`. Even though users `U0, U1, U2` have access to read this dataset, only `U0` and `U1` can decrypt its encrypted patterns.

Key Class	Dataset
`keyclass: 0` Keys: `keyId: 0, keyId: 1` Users: `U0, U1`	`datasetA` Users: `U0, U1, U2`

User `U1` has been given access to an **additional** keyclass, `1`, which contains key IDs `11` and `22`. These keys are used to encrypt certain **other** patterns in `datasetA`. Even though users `U0, U1, U2` have access to read this dataset – same as above – only `U1` can decrypt the additional encrypted patterns.

Key Class	Dataset
`keyclass: 1` Keys: `keyId: 11, keyId: 22` Users: `U1`	`datasetA` Users: `U0, U1, U2`

Configuring Keys with the CLI

When using the `local` key management system, encryption keys in Cribl Stream are encrypted with `$CRIBL_HOME/local/cribl/auth/cribl.secret` and stored in `$CRIBL_HOME/local/cribl/auth/keys.json`. Cribl monitors the `keys.json` file for changes every 60 seconds.

When installed as a Splunk app, `$CRIBL_HOME` is `$SPLUNK_HOME/etc/apps/cribl`.

Listing Keys

Keys are added and listed using the `keys` command:

```
$CRIBL_HOME/bin/cribl keys list -g <workerGroupID>
```

Sample Command Output

```
keyId  description  algorithm    keyclass  kms    created         expires     useI\
-----------------------------------------------------------------------------------
1      cbc          aes-256-cbc  0         local  1544906269.316  0           fals
2      gcm          aes-256-gcm  1         local  1544906272.452  0           fals
3      cbc          aes-256-cbc  2         local  1544906275.948  1545906275  true
4      gcm          aes-256-gcm  3         local  1544906278.026  0           true
```

Adding Keys

Displaying `--help`:

```
$CRIBL_HOME/bin/cribl keys add --help
```

Sample Command Output

```
Add encryption keys
Usage: [options] [args]

Options:
[-a <algorithm>] - Encryption algorithm. Supported values: aes-256-cbc (default), a
[-c <keyclass>]  - Key class to set for the key.
[-k <kms>]       - KMS to use. Must be configured; see cribl.yml.
[-e <expires>]   - Expiration time, in epoch time.
[-i]             - Use an initialization vector.
 -g <group>      - Worker Group's ID.
```

Adding a key to keyclass `1`, with no expiration date, on the `default` Worker Group:

```
$CRIBL_HOME/bin/cribl keys add -c 1 -i -g default
```

Sample Command Output

```
Adding key: success. Key count=1
```

(You would use the same syntax to reference a non-`default` Worker Group by its name.) ? Listing keys to verify key generation:

```
$CRIBL_HOME/bin/cribl keys list -g default
```

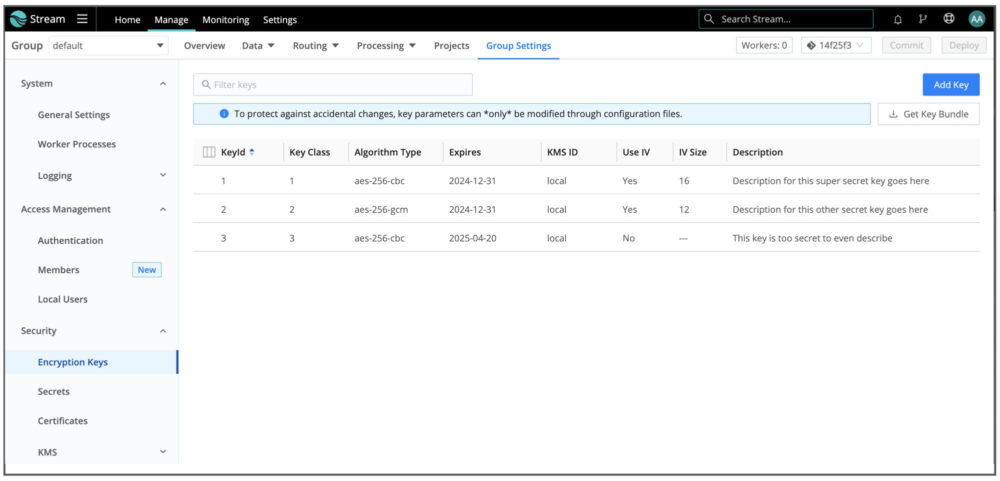
Sample Command Output

```
keyId  algorithm    keyclass  kms    created          expires     useIV
-----------------------------------------------------------------------
1      aes-256-cbc  1         local  1545243364.342  0           true
```

Configuring Keys with the UI

In a single-instance deployment, you can access the key management interface through **Settings** > **Security** > **Encryption Keys**. In a distributed deployment, for each Group, select **Manage** > `<group-name>` > **Group Settings** > **Security** > **Encryption Keys**.

The resulting **Encryption Keys** page lists existing keys, with sortable columns.

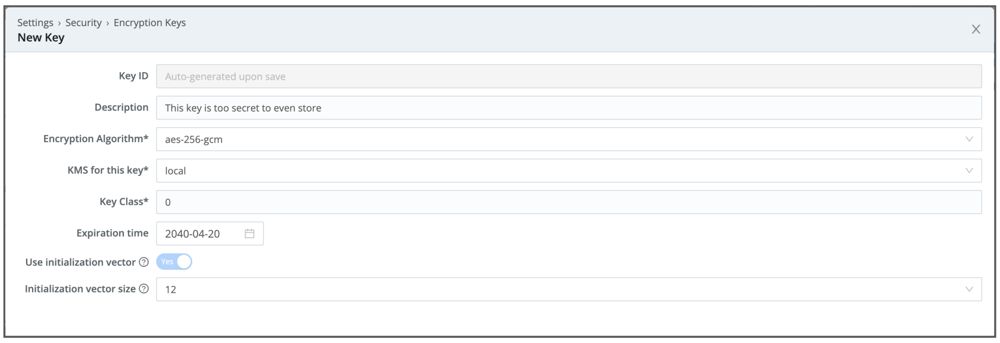


To protect against accidental changes, key parameters can *only* be modified through configuration files.

KeyId	Key Class	Algorithm Type	Expires	KMS ID	Use IV	IV Size	Description
1	1	aes-256-cbc	2024-12-31	local	Yes	16	Description for this super secret key goes here
2	2	aes-256-gcm	2024-12-31	local	Yes	12	Description for this other secret key goes here
3	3	aes-256-cbc	2025-04-20	local	No	---	This key is too secret to even describe

List or add keys through Cribl Stream's UI

Click **Add Key** to display the creation modal shown below.



Key ID	Auto-generated upon save
Description	This key is too secret to even store
Encryption Algorithm*	aes-256-gcm
KMS for this key*	local
Key Class*	0
Expiration time	2040-04-20
Use initialization vector	Yes
Initialization vector size	12

New Key configuration modal

Modifying Keys

To protect against accidental changes, a key's parameters – once saved – can be edited only through configuration files.

When you update keys by editing the `keys.json` file, you must add them back to the directories above (respectively, on a single instance or on a distributed deployment's Leader Node).

Sync `auth/(cribl.secret|keys.json)`

To successfully decrypt data, the `decrypt` command will need access to the same keys that were used to encrypt, **in the Cribl instance where encryption happened**.

- In a single-instance deployment, the `cribl.secret` and `keys.json` files reside in: `$CRIBL_HOME/local/cribl/auth/`.
- In a distributed deployment, the `cribl.secret` and `keys.json` files reside on the Leader Node in: `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.
- When using the UI, you can download these files by clicking the **Get Key Bundle **button.

Sync/copy these files over to their counterparts on the Search Head/decrypting side, residing in: `$SPLUNK_HOME/etc/apps/cribl/local/cribl/auth/`.

11.6.2. Decryption

Decryption of Data

Currently, Cribl Stream supports decryption only when Splunk is the end system. In Splunk, decryption is available to users of any role with permissions to run the `cribldecrypt` command that ships with Cribl App for Splunk. Further restrictions can be applied with Splunk **capabilities**. This page provides details.

As of v.3.5.3, Cribl has added `cribldecrypt` as an alias to the original `decrypt` command. Use this alias to avoid conflicts with Splunk's internal commands. (We show it in the examples below.) Both are, in fact, aliases to the actual command: `/path/2/cribl --splunk-decrypt`. You can use both aliases.

Usage

The `cribldecrypt` command is used to display Cribl-encrypted fields in cleartext. It is an alias to the `decrypt` command. This command decrypts fields only for the encryption keys that the user can access.

The example below retrieves data from a Splunk index with Cribl-encrypted data, and pipes it to the `cribldecrypt` command:

```
index=index_with_encrypted_fields | cribldecrypt
```

Decrypting in Splunk

Decryption in Splunk is implemented via a custom command called `cribldecrypt`. To use the command, users must belong to a Splunk role that has permissions to execute it. Capabilities, which are aligned to Cribl Key Classes, can be associated with a particular role to further control the scope of `cribldecrypt`.

Decrypt Command Is Search Head ONLY

To ensure that keys don't get distributed to all search peers – including peers that your search head can search, but you don't have full control over – `cribldecrypt` is scoped to run locally on the installed search head.

Restricting Access with Splunk Capabilities

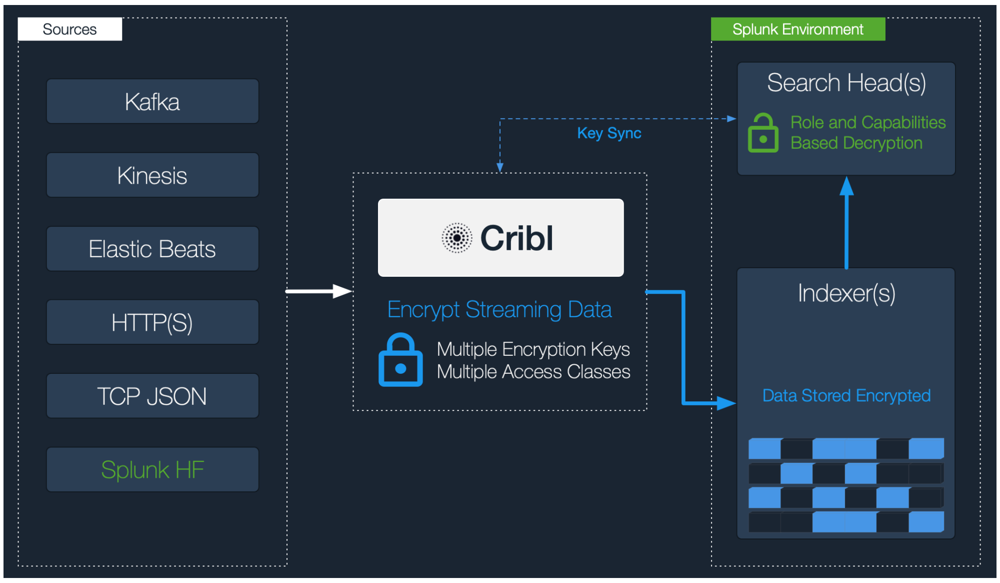
In Splunk, capability names should follow the format `cribl_keyclass_N`, where `N` is the Cribl Key Class. For example, a role with capability `cribl_keyclass_1` has access to all key IDs associated with key class `1`.

Capability Name	Corresponding Cribl Key Class

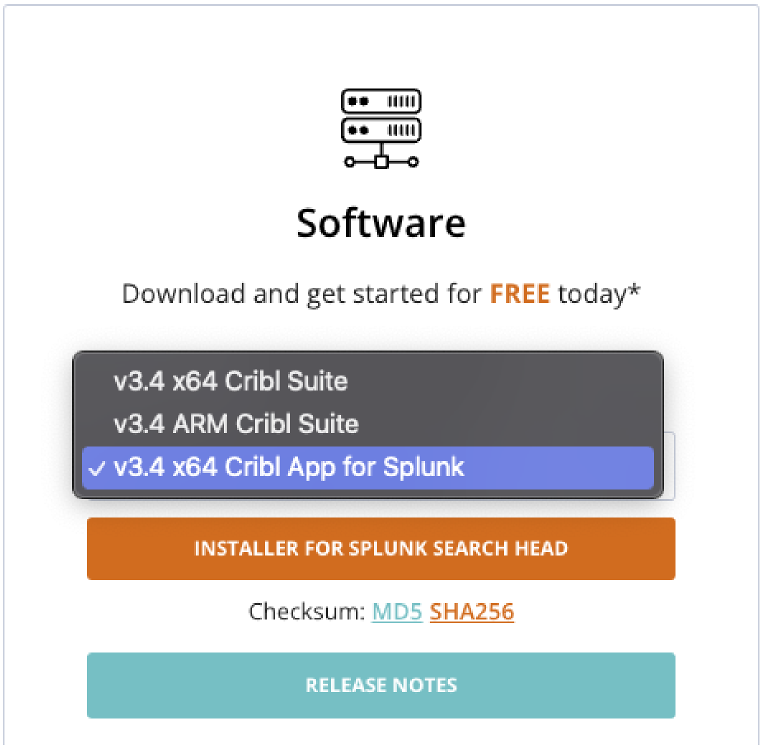
`cribl_keyclass_1`
`cribl_keyclass_2`
…
`cribl_keyclass_N`
	1
	2
	…
	N

Configuring Splunk Search Head to Decrypt Data

You set up decryption in Splunk according to this schematic:



1. Download the Cribl Stream App for Splunk from Cribl's Download Cribl Stream page: In the **On Prem** section, select the Splunk app from the drop-down list, as shown. Clicking the orange button downloads a file named: `cribl-splunk-app-<version-#>-<hash-#>-linux-x64.tgz`.


Software
Download and get started for FREE today*
v3.4 x64 Cribl Suite
v3.4 ARM Cribl Suite
✓ v3.4 x64 Cribl App for Splunk
INSTALLER FOR SPLUNK SEARCH HEAD
Checksum: MD5 SHA256
RELEASE NOTES

Downloading Cribl's Splunk app

2. To install the Cribl/LogStream App for Splunk on your search head, untar the package into your `$SPLUNK_HOME/etc/apps` directory.
   The app will run in search head mode by default. If the app has previously been installed and later modified, you can convert it to search head mode with the command: `$CRIBL_HOME/bin/cribld mode-searchhead`. (When installed as a Splunk app, `$CRIBL_HOME` is `$SPLUNK_HOME/etc/apps/cribl`.)
3. Assign permissions to the `cribldecrypt` command, per your requirements.
4. Assign capabilities to your roles, per your requirements. If you'd like to create more capabilities, ensure that they follow the naming convention defined above.
5. In the `$SPLUNK_HOME/etc/apps/cribl/local/cribl/auth/` directory, sync `cribl.secret` | `keys.json`. (To successfully decrypt data, the `cribldecrypt` command will need access to the same keys that were used to encrypt, **in the Cribl instance where encryption happened**.)

- In a single-instance deployment, the `cribl.secret` and `keys.json` files reside in: `$CRIBL_HOME/local/cribl/auth/`.
- In a distributed deployment, these files reside on the Leader Node in: `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.
- When using Cribl Stream's UI, you can download these files by clicking the **Get Key Bundle **button.

Sync/copy these files over to their counterparts on the search head (decryption side). In a non-Splunk integration, you would copy these assets to wherever decryption will take place.

Modifying Keys

When you update keys by editing the `keys.json` file, you must add them back to the directories above (respectively, on a single instance or on a distributed deployment’s Leader Node).

11.7. Usage Examples

11.7.1. Encrypting Sensitive Data

Encryption at Ingest-Time and Decryption in Splunk

With Cribl Stream, you can encrypt your sensitive data in real time before it's forwarded to and stored at a destination. Using the out-of-the-box Mask function, you can define patterns to encrypt with specific key IDs or key classes. To decrypt in Splunk, you will need to install Cribl App for Splunk on your search head. (The app will default to `mode-searchhead`.)

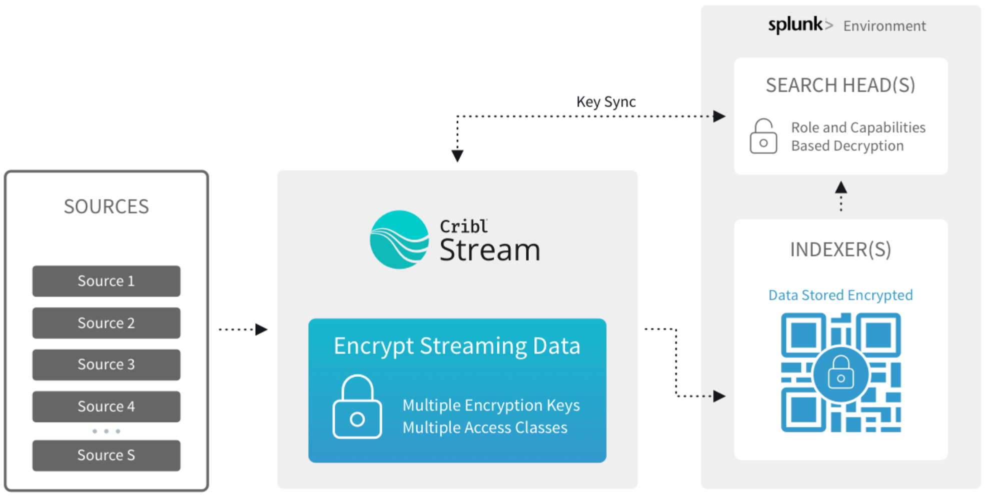
Keys and Key Classes

Symmetric encryption keys can be configured through the CLI or the UI. They're used to encrypt the patterns, and users are free to define as many keys as required.

Key classes are collections of keys that can be used to implement multiple levels of access control. Users (or groups of users) that have access to data with encrypted patterns can be associated with key classes. You can use these classes to provide more-granular access rights, such as read versus decryption permissions on a dataset.

Encrypting in Cribl Stream and Decrypting in Splunk

1. Define one or more keys and key classes on Cribl Stream. (See UI- and CLI-based instructions.)
2. Sync `auth` with the decryption side (Splunk Search Head). (The Splunk-side directory is `$SPLUNK_HOME/etc/apps/cribl/local/cribl/auth/`.)
3. Apply the Mask function to patterns of interest, using C.Crypto.encrypt().
4. Decrypt on the Splunk search head, using Role-Based Access Control on the `decrypt` command.



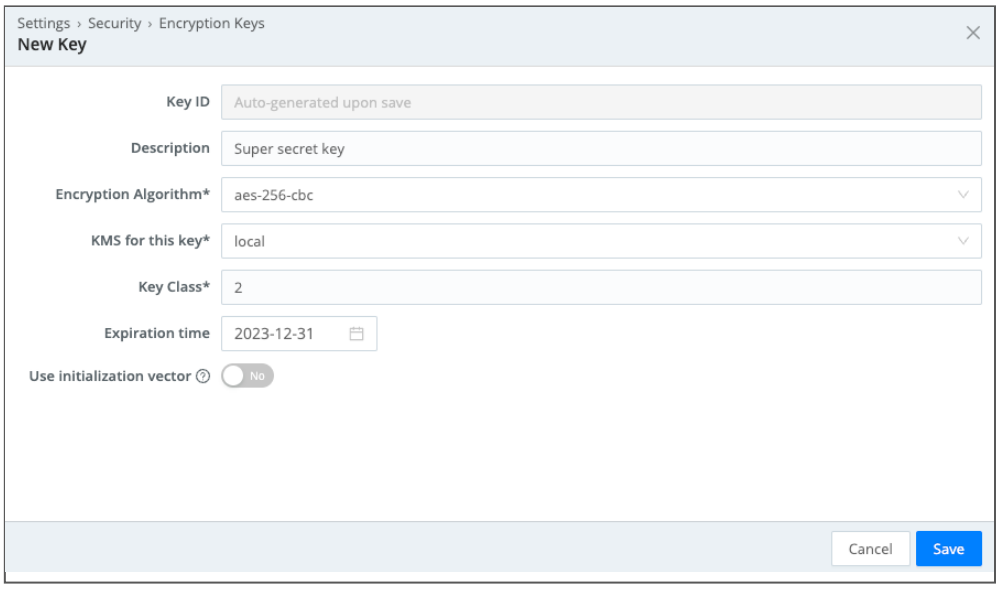
Encrypting in Cribl Stream, decrypting in Splunk

Examples

Encryption Side

You can generate keys via the UI or the CLI.

To generate keys via the UI, access **Group Settings** > **Security > Encryption Keys**:


Settings › Security › Encryption Keys
New Key

Key ID	Auto-generated upon save
Description	Super secret key
Encryption Algorithm*	aes-256-cbc
KMS for this key*	local
Key Class*	2
Expiration time	2023-12-31
Use initialization vector	No

Adding a new encryption key

To generate one or more keys via the CLI, pattern your commands after these examples.

In a single-instance deployment:

```
$CRIBL_HOME/bin/cribl keys add -c 1 -i ... $CRIBL_HOME/bin/cribl keys add -c <N> -i
```

In a distributed deployment, to generate keys on a Worker Group named `uk`:

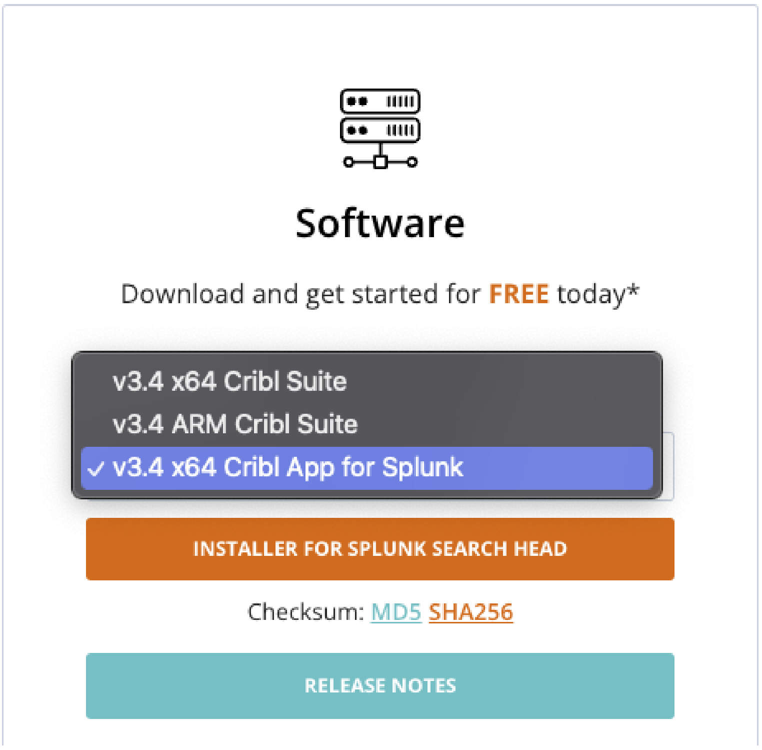
```
$CRIBL_HOME/bin/cribl keys add -c 1 -i -g uk ... $CRIBL_HOME/bin/cribl keys add -c
<N> -i -g uk
```

Add `-e <epoch>` to the above commands if you'd like to set expiration for your keys.

For all command/syntax options, see Adding Keys.

Decryption Side

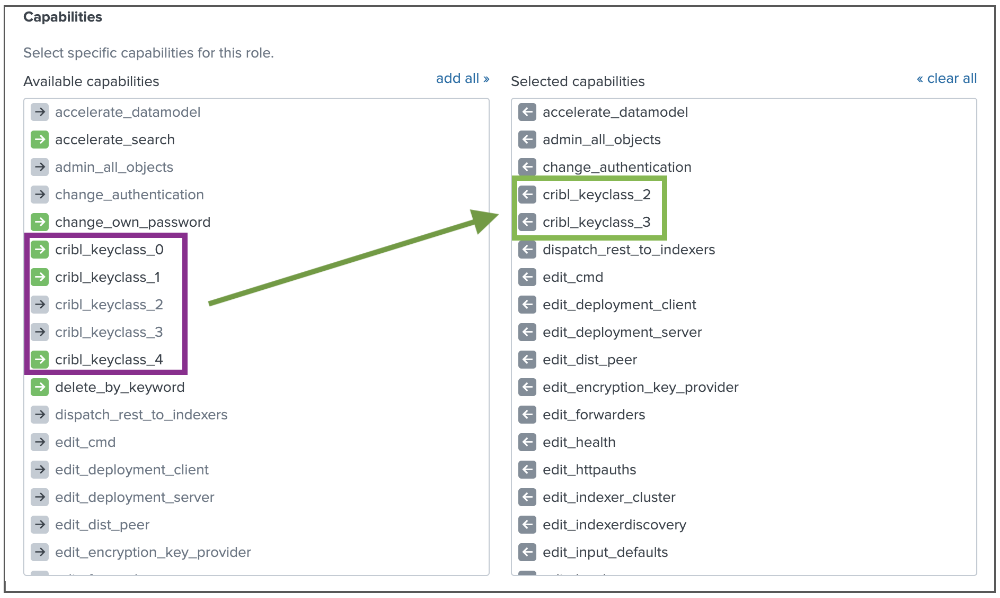
- Download the Cribl Stream App for Splunk from Cribl's Download Cribl Stream page: In the **On Prem** section, select the Splunk app from the drop-down list, as shown. Clicking the orange button downloads a file named: `cribl-splunk-app-<version-#>-<hash-#>-linux-x64.tgz`.


Software
Download and get started for FREE today*
v3.4 x64 Cribl Suite
v3.4 ARM Cribl Suite
✓ v3.4 x64 Cribl App for Splunk
INSTALLER FOR SPLUNK SEARCH HEAD
Checksum: MD5 SHA256
RELEASE NOTES

Downloading Cribl's Splunk app

- To install the Cribl Stream App for Splunk on your search head, untar the package into your `$SPLUNK_HOME/etc/apps` directory. The app will default to `mode-searchhead`.
- Assign permissions to the `decrypt` command, per your requirements.
- Assign capabilities to your Roles, per your requirements. Capability names should follow the format `cribl_keyclass_N`, where `N` is the Cribl Key Class. For example, a role with capability

`cribl_keyclass_1` has access to all key IDs associated with key class `1`. You can use more capabilities, as long as they follow this naming convention.



Selecting capabiities

- In the `$SPLUNK_HOME/etc/apps/cribl/local/cribl/auth/` directory, sync `cribl.secret` | `keys.json`. (To successfully decrypt data, the `decrypt` command will need access to the same keys that were used to encrypt, **in the Cribl instance where encryption happened**.)
    - In a single-instance deployment, the `cribl.secret` and `keys.json` files reside in: `$CRIBL_HOME/local/cribl/auth/`.
    - In a distributed deployment, these files reside on the Leader Node in: `$CRIBL_HOME/groups/<group-name>/local/cribl/auth/`.
    - When using Cribl Stream's UI, you can download these files by clicking **Get Key Bundle**.

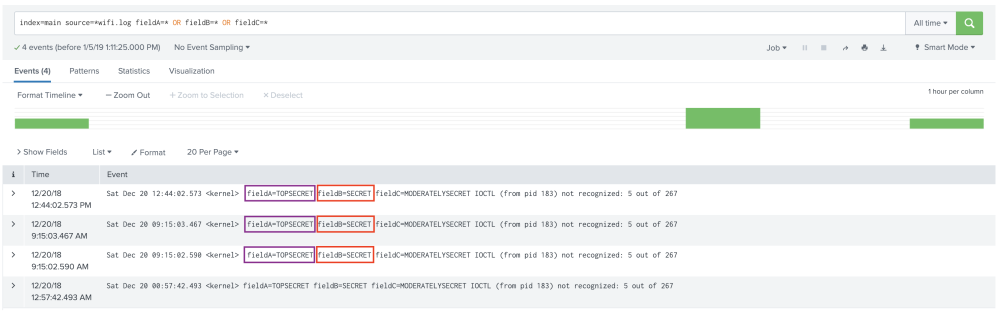
Sync/copy these files over to their counterparts on the search head (decryption side). In a non-Splunk integration, you would copy these assets to wherever decryption will take place.

Modifying Keys

When you update keys by editing the `keys.json` file, you must add them back to the directories above (respectively, on a single instance or on a distributed deployment's Leader Node).

Usage

Before Encryption: Sample un-encrypted events. Notice the values of `fieldA` and `fieldB`.



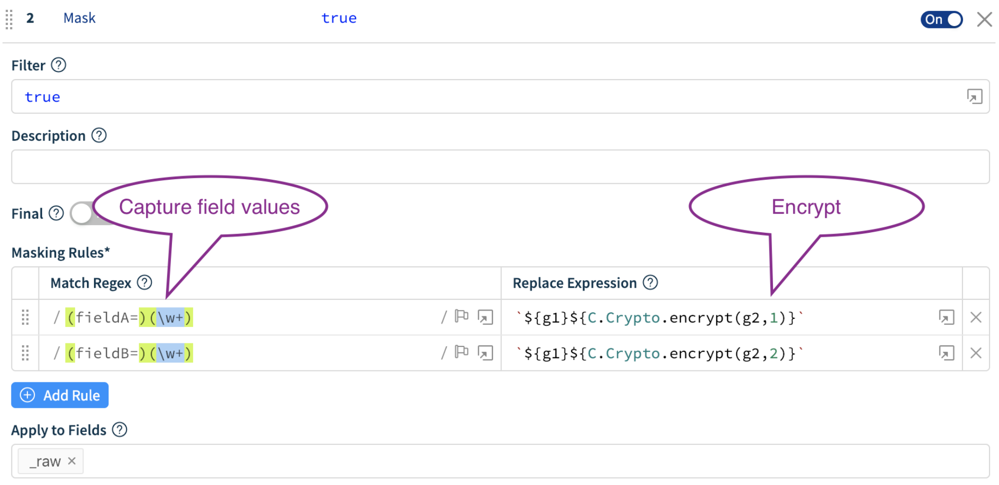
```
index=main source=*wifi.log fieldA=* OR fieldB=* OR fieldC=*
```

✓ 4 events (before 1/5/19 1:11:25.000 PM)

Time	Event
12/20/18 12:44:02.573 PM	`Sat Dec 20 12:44:02.573 <kernel> fieldA=TOPSECRET fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:03.467 AM	`Sat Dec 20 09:15:03.467 <kernel> fieldA=TOPSECRET fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:02.590 AM	`Sat Dec 20 09:15:02.590 <kernel> fieldA=TOPSECRET fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 12:57:42.493 AM	`Sat Dec 20 00:57:42.493 <kernel> fieldA=TOPSECRET fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`

Events before encryption

Next, encrypt `fieldA` values with key class `1`, and `fieldB` with key class `2`.



2 Mask `true` On

Filter: `true`

Description:

Final:

Masking Rules*

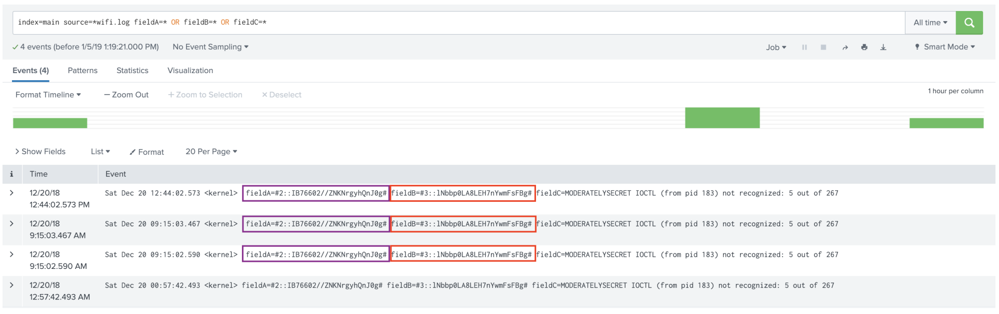
Match Regex	Replace Expression
`/(fieldA=)(\w+)/`	`` `${g1}${C.Crypto.encrypt(g2,1)}` ``
`/(fieldB=)(\w+)/`	`` `${g1}${C.Crypto.encrypt(g2,2)}` ``

Add Rule

Apply to Fields: `_raw`

Encrypting two fields with separate key classes

After Encryption: again, notice the values of `fieldA` and `fieldB`.



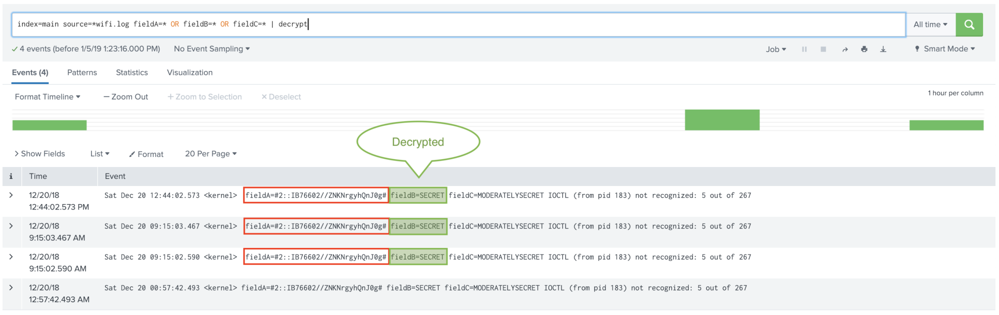
```
index=main source=*wifi.log fieldA=* OR fieldB=* OR fieldC=*
```

✓ 4 events (before 1/5/19 1:19:21.000 PM)

Time	Event
12/20/18 12:44:02.573 PM	`Sat Dec 20 12:44:02.573 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=#3::lNbbp0LA8LEH7nYwmFsFBg# fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:03.467 AM	`Sat Dec 20 09:15:03.467 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=#3::lNbbp0LA8LEH7nYwmFsFBg# fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:02.590 AM	`Sat Dec 20 09:15:02.590 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=#3::lNbbp0LA8LEH7nYwmFsFBg# fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 12:57:42.493 AM	`Sat Dec 20 00:57:42.493 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=#3::lNbbp0LA8LEH7nYwmFsFBg# fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`

Both fields encrypted

Here, we've decrypted `fieldB` but not `fieldA`. This is because the logged-in user has been assigned the capability `cribl_keyclass_2`, but not `cribl_keyclass_1`.



```
index=main source=*wifi.log fieldA=* OR fieldB=* OR fieldC=* | decrypt
```

Time	Event
12/20/18 12:44:02.573 PM	`Sat Dec 20 12:44:02.573 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:03.467 AM	`Sat Dec 20 09:15:03.467 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 9:15:02.590 AM	`Sat Dec 20 09:15:02.590 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`
12/20/18 12:57:42.493 AM	`Sat Dec 20 00:57:42.493 <kernel> fieldA=#2::IB76602//ZNKNrgyhQnJ0g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267`

One field decrypted

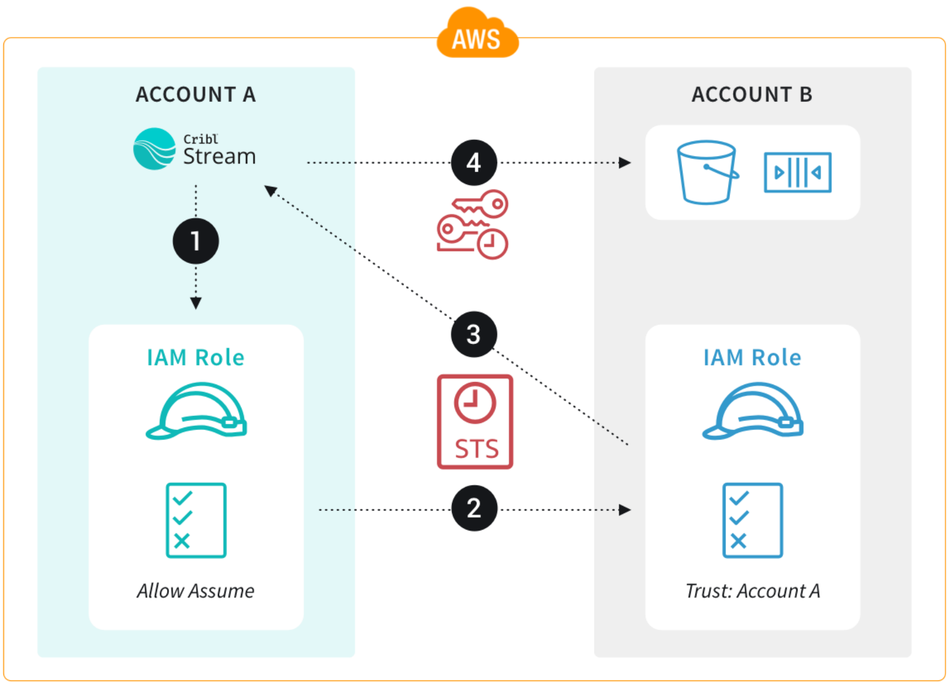
11.7.2. AWS Cross-Account Data Collection

Cribl uses multiple AWS accounts for different purposes, including our public Cribl.Cloud service (deployment details here). We built our account strategy from the ground up using the AWS Control Tower framework, as summarized in our Logging in a Multi-Account AWS Environment blog post.

However, some organizations use a legacy account structure that doesn't consolidate logs to a single account. This creates a challenge in collecting data or logs from peer accounts, as permission boundaries now need to be crossed. Whether you need to collect data from, or write data to, another account, the `AssumeRole` permission allows for cross-account access without the need to generate static IAM keys.

AssumeRole Example

A usage example is a central logging account that has access to other organization accounts to consume logs, but not to perform other actions. Let's say we have two AWS accounts, A and B. We want account A to be able to access resources in account B. We can build a policy inside account B that allows permissions to access the target resources. We can then specify, in account B, that we trust account A to be allowed to use this role. The diagram below illustrates how the `AssumeRole` action permits the Trusted Account (A) access to resources in the Trusting Account (B).



Using STS and temporary credentials to access an S3 bucket and SQS queue across accounts

Here's how Cribl Stream would work with `AssumeRole` permissions inside AWS:

1. The Cribl Stream Worker has an EC2 instance role attached.
2. The IAM role in Account A permits the EC2 instance to assume the role in Account B (and Account B trusts Account A).
3. Temporary IAM credentials are returned to the EC2 instance.
4. Cribl Stream uses the temporary IAM credentials to access the resources in Account B.

Since this guide's original publication, we've updated `logstream` to `cribl-stream` in example ARNs below, to match our renamed product.

Configure IAM AssumeRole Permissions

First, in account A, we build a policy that allows only the ability to assume the role inside account B. This policy restricts users from being able to access any resources that they don't need to see. We can also revoke this trust relationship at any time, without having to worry about an account still having keys and (therefore) access to the data.

In our example, we want to access VPC Flow logs inside an S3 bucket in account B (ID 222222222222) from account A (ID 111111111111). We'll start building the two policies in account B, and then move to account A.

Account B Configuration

In account B, we build a policy to enable access to the S3 bucket (`vpc-flow-logs-for-cribl`) with the least privileges required for Cribl Stream. This policy is the one that changes depending on what you need to accomplish – e.g., reading from an S3 bucket, writing to Kinesis Streams, etc.

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Action": "s3:GetObject",
     "Resource": "arn:aws:s3:::vpc-flow-logs-for-cribl/*"
   },
   {
     "Effect": "Allow",
     "Action": "s3:ListBucket",
     "Resource": "arn:aws:s3:::vpc-flow-logs-for-cribl"
   }
 ]
}
```

Next, we need to attach a Trust Relationship to Account B's IAM role to permit the `AssumeRole` action from account A:

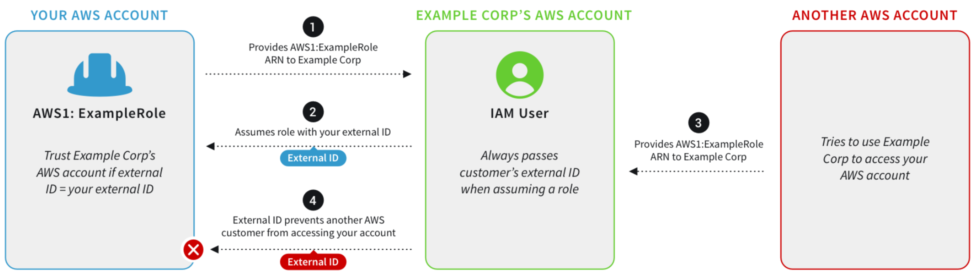
```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111111111111:role/account-a-cribl-stream-assumerole-ro
     },
     "Action": "sts:AssumeRole",
     "Condition": {
       "StringEquals": {
         "sts:ExternalId": "cribl-s3cre3t"
       }
     }
   }
 ]
}
```

If you are creating a Trust Relationship for a Cribl.Cloud Worker managed by Cribl, you would instead paste the AWS Worker's role ARN in the format shown below. This option applies only to Cribl-managed Workers, not to hybrid Workers on customer-managed Cribl Stream instances:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "AWS": "arn:aws:iam::111111111111:role/main-default"
      },
      "Action": "sts:AssumeRole",
      "Condition": {
        "StringEquals": {
          "sts:ExternalId": "cribl-s3cre3t"
        }
      }
    }
  ]
}
```

Why an External ID Condition in the Trust Policy?

It is important to configure an AWS External ID, especially if you have third parties accessing your AWS accounts. The External ID protects from the confused deputy problem, where a third party obtains access through an intermediary. The External ID is not a password or secret, but it should still be protected from accidental sharing.


YOUR AWS ACCOUNT
AWS1: ExampleRole
Trust Example Corp's AWS account if external ID = your external ID
1
Provides AWS1:ExampleRole ARN to Example Corp
2
Assumes role with your external ID
External ID
4
External ID prevents another AWS customer from accessing your account
External ID
EXAMPLE CORP'S AWS ACCOUNT
IAM User
Always passes customer's external ID when assuming a role
3
Provides AWS1:ExampleRole ARN to Example Corp
ANOTHER AWS ACCOUNT
Tries to use Example Corp to access your AWS account

Mitigating the confused deputy vulnerability

Account A Configuration

In account A, we next configure a new IAM role with the following policy.

If you are creating a Trust Relationship for a Cribl-managed Cribl.Cloud Worker, skip this section. Cribl has implicitly preconfigured Account A for you.

For our example, we only want the role to be able to use the `AssumeRole` action, but you can add additional statements to meet your needs:

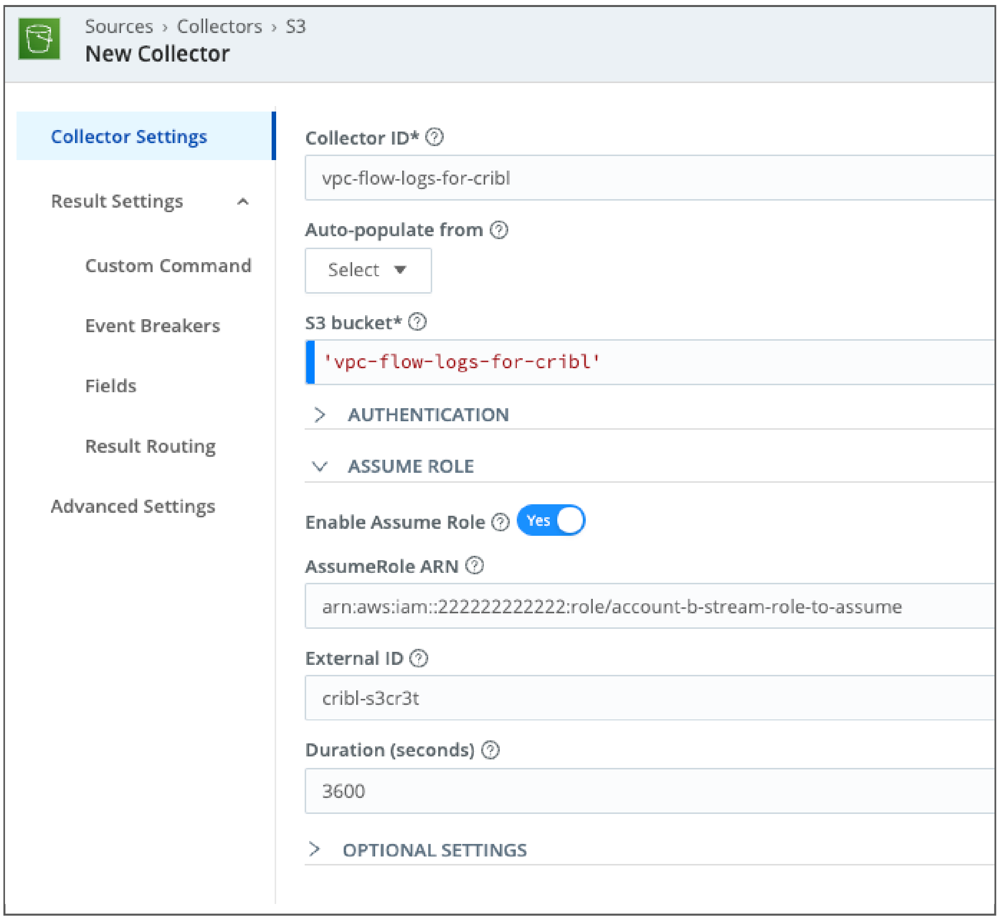
```
{
 "Version": "2012-10-17",
 "Statement": {
   "Effect": "Allow",
   "Action": "sts:AssumeRole",
   "Resource": "arn:aws:iam::222222222222:role/account-b-cribl-stream-role-to-assur
 }
}
```

Since we need our EC2 instance to be able to assume this role, we will configure the Trust Relationship as follows:

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "Service": "ec2.amazonaws.com"
     },
     "Action": "sts:AssumeRole"
   }
 ]
}
```

Configure Cribl Stream

Now that our `AssumeRole` policies have been built, we can configure Cribl Stream to assume the correct role to access the resources we need. Configure your Source, Collector, or Destination with the appropriate `AssumeRole` ARN and External ID. While this screenshot shows an S3 collector specifically, all AWS sources and destinations support Assume Role functionality.



Collector ID*: vpc-flow-logs-for-cribl

Auto-populate from: Select

S3 bucket*: `'vpc-flow-logs-for-cribl'`

Enable Assume Role: Yes

AssumeRole ARN: arn:aws:iam::222222222222:role/account-b-stream-role-to-assume

External ID: cribl-s3cr3t

Duration (seconds): 3600

Cribl Stream Assume Role configuration

11.7.3. SELinux Configuration

This page explains how to make Cribl Stream work correctly with SELinux in **enforcing mode**.

If you're unsure about SELinux and its modes, here's some background:

- Security-Enhanced Linux (SELinux) is a mandatory access control (MAC) security mechanism implemented in the kernel. (Source: CentOS Wiki.)
- SELinux defines access controls for the applications, processes, and files on a system. It uses security policies, which are a set of rules that tell SELinux what can or can't be accessed, to enforce the access allowed by a policy. (Source: Red Hat.)
- Processes and files are labeled with an **SELinux context** that contains additional information, such as an SELinux user, role, type, and, optionally, a level. When running SELinux, all of this information is used to make access control decisions. (Source: Red Hat.)
- SELinux has two modes, **enforcing** and **permissive**. When running in enforcing mode, SELinux enforces the SELinux policy and denies access based on SELinux policy rules. (Source: Red Hat.)

SELinux and Cribl Stream

Cribl Stream is typically used for in-stream processing, where the processes and files that Cribl Stream Workers need to access are known beforehand. This makes it easy to write an SELinux policy that's relatively restrictive of processes and files.

For network ports, though, the policy should be more permissive, because users can configure those within the UI.

SELinux Troubleshooting

SELinux problems usually surface when you try to start the Cribl Stream service. Users immediately receive an error like the following:

Cribl Stream fails to start

```
Job for cribl.service failed because the control process exited with error code.
See "systemctl status cribl.service" and "journalctl -xe" for details.
```

One indication that the problem is in SELinux configuration, not in the Cribl Stream service itself, is when (1) nothing has been logged in `$CRIBL_HOME/log/cribl.log`, and (2), when you run `journalctl -xe` you see errors like the following:

Errors from journalctl -xe

```
cribl.service: Failed to execute command: Permission denied
cribl.service: Failed at step EXEC spawning /opt/cribl/bin/cribl: Permission denie

cribl.service: Control process exited, code=exited status=203
cribl.service: Failed to execute command: No such file or directory
cribl.service: Failed at step EXEC spawning /bin/rm -f /opt/cribl/pid/cribl.pid: Pe
```

You can confirm that SELinux configuration is the issue by verifying that the Cribl Stream service is able to start up on its own. To do this, run the `cribl start` command.

Here are three commands that you can think of as a basic SELinux troubleshooting toolkit:

- `getenforce` verifies that SELinux is enabled and enforcing policy.
- `sestatus -v` shows details of the SELinux policy that's in effect.
- `ls` with the `-Z` or `--context` option shows the SELinux tags on files and/or folders. In the output, the fifth column reports the SELinux context in the form `user:role:type:level`.

Example Problem

Suppose you run `ls -lZ /opt/cribl` to see the SElinux context for `/opt/cribl`.

If the output looks like the following, there is a problem: the presence of "home" among the `type` part of the Linux context. In this case, `user_home_t` is the offending type. When the type is `user_home_t` or `admin_home_t`, SELinux will not permit the service to read, write, or execute files.

Problematic SELinux context output

```
$ ls -laZ /opt/cribl
total 0
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:user_home_t:s0  62 Feb 16 08:31 .
drwxr-xr-x. 3 root  root  system_u:object_r:usr_t:s0            19 Feb 16 19:54 ..
drwxr-xr-x. 2 cribl cribl unconfined_u:object_r:user_home_t:s0 202 Feb 16 08:31 bi
drwxr-xr-x. 5 cribl cribl unconfined_u:object_r:user_home_t:s0  49 Feb 16 08:20 da
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:user_home_t:s0  59 Feb 16 08:31 de
drwxr-xr-x. 3 cribl cribl unconfined_u:object_r:user_home_t:s0  22 Feb 16 08:31 th
```

Example Solution

To fix the labeling, download Cribl Stream again directly to `/opt`. **Do not** download Cribl Stream to a user's home directory, or to root (`/`), and then move it to `/opt` - that was what caused our example problem.

Downloading Cribl Stream to /opt

```
curl -Lso - $(curl -s https://cdn.cribl.io/dl/latest) | sudo tar zxvf - -C /opt
```

A simpler option is to revert the labels using restorecon, an SELinux policy utility that's in the `policycoreutils-python-utils` package.

Reverting labels with restorecon

```
restorecon -R -v /opt/cribl
systemctl restart cribl
```

Once you correct the problem, the output will look like the following. Instead of `*home_t` listings in the `type` field, this SELinux context has `usr_t`, a type that will satisfy SELinux's criteria for allowing the service to read, write, and execute files.

Corrected SELinux context output

```
# ls -laZ /opt/cribl
total 0
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:usr_t:s0  62 Feb 16 08:31 .
drwxr-xr-x. 3 root  root  system_u:object_r:usr_t:s0      19 Feb 16 20:05 ..
drwxr-xr-x. 2 cribl cribl unconfined_u:object_r:usr_t:s0 202 Feb 16 08:31 bin
drwxr-xr-x. 5 cribl cribl unconfined_u:object_r:usr_t:s0  49 Feb 16 08:20 data
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:usr_t:s0  59 Feb 16 08:31 default
drwxr-xr-x. 3 cribl cribl unconfined_u:object_r:usr_t:s0  22 Feb 16 08:31 thirdpart
```

12. Monitoring

To get an operational view of a Cribl Stream deployment, you can consult the following resources.

Monitoring Resources

- Monitoring Page
- Internal Logs and Metrics
- Licensing
- Flows
- Health Endpoint

Monitoring Page

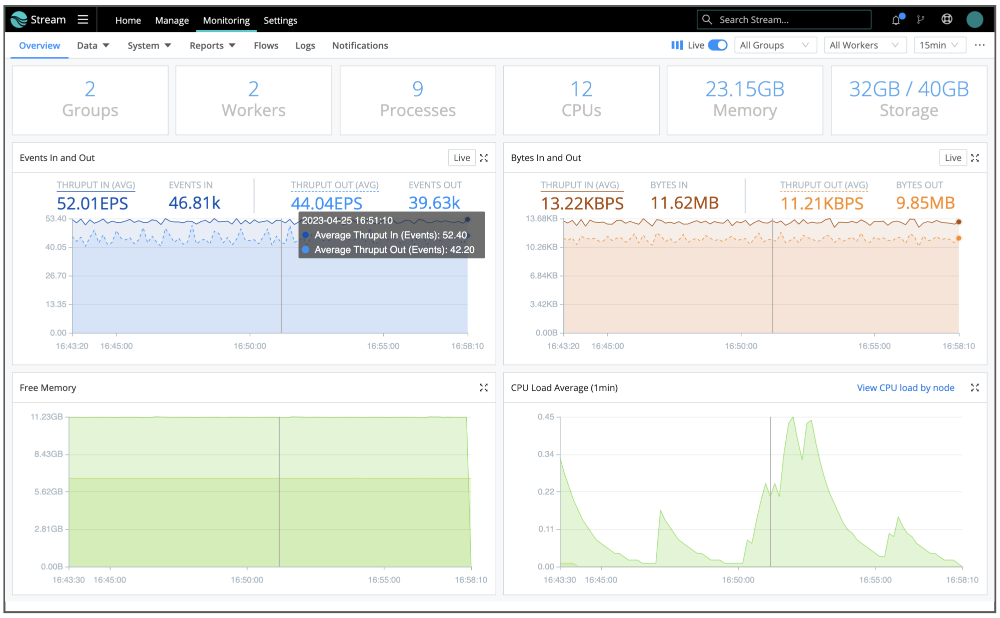
From Cribl Stream's top nav, select **Monitoring** to access multiple dashboards. The resulting **Monitoring** page displays information about traffic in and out of the system, as well as collection jobs and tasks. It tracks events, bytes, splits by data fields over time, and broader system metrics.

The initial view (below) shows aggregate data for all Worker Groups and all Workers. You can use the drop-downs at the upper right to isolate individual Groups, or individual Workers.

Also at the upper right, you can change the display's granularity from the default `15 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.)

The displayed **Storage** tile (upper right) 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 Worker or Group. 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.



Monitoring page

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

Byte-related charts show the uncompressed amounts and rates of data processed over the selected time range:

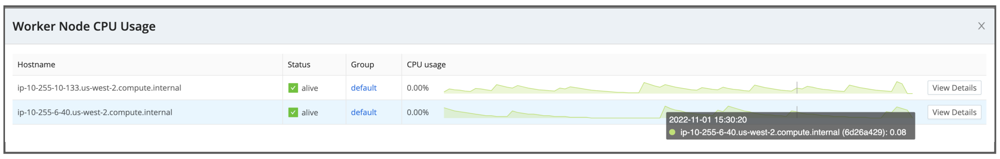
- Events (total) in and out
- Events per second in and out
- Bytes (total) in and out
- Bytes per second in and out

We measure bytes in 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. Bytes out is measured by the full serialized size of the payload.

The displayed **CPU Load Average** is an average per Worker Process, updated at 1-minute granularity. (It is not an average for the Worker Node as a whole.)

Vertical lines across each chart display configuration changes. Click anywhere on the line to view summary information including time, data, and configuration versions.

Select **View CPU load by node** to open a modal showing CPU usage per Worker.

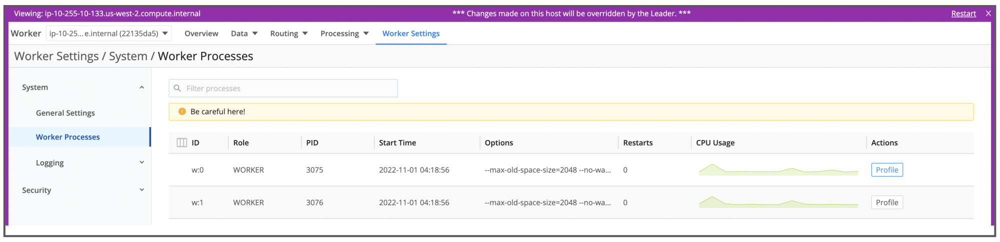


Worker Node CPU Usage

Hostname	Status	Group	CPU usage	
ip-10-255-10-133.us-west-2.compute.internal	✓ alive	default	0.00%	View Details
ip-10-255-6-40.us-west-2.compute.internal	✓ alive	default	0.00%	View Details

Monitoring page

Select a **View Details** button to access a Worker Node's **Worker Settings**, with details per Worker Process. (To enable this button, you must first enable the parent Group's Remote UI Access.)


Viewing: ip-10-255-10-133.us-west-2.compute.internal
*** Changes made on this host will be overridden by the Leader. ***
Restart
Worker
ip-10-25... e.internal (22135da5)
Overview
Data
Routing
Processing
Worker Settings
Worker Settings / System / Worker Processes
System
General Settings
Worker Processes
Logging
Security
Filter processes
Be careful here!

ID	Role	PID	Start Time	Options	Restarts	CPU Usage	Actions
w:0	WORKER	3075	2022-11-01 04:18:56	--max-old-space-size=2048 --no-wa...	0		Profile
w:1	WORKER	3076	2022-11-01 04:18:56	--max-old-space-size=2048 --no-wa...	0		Profile

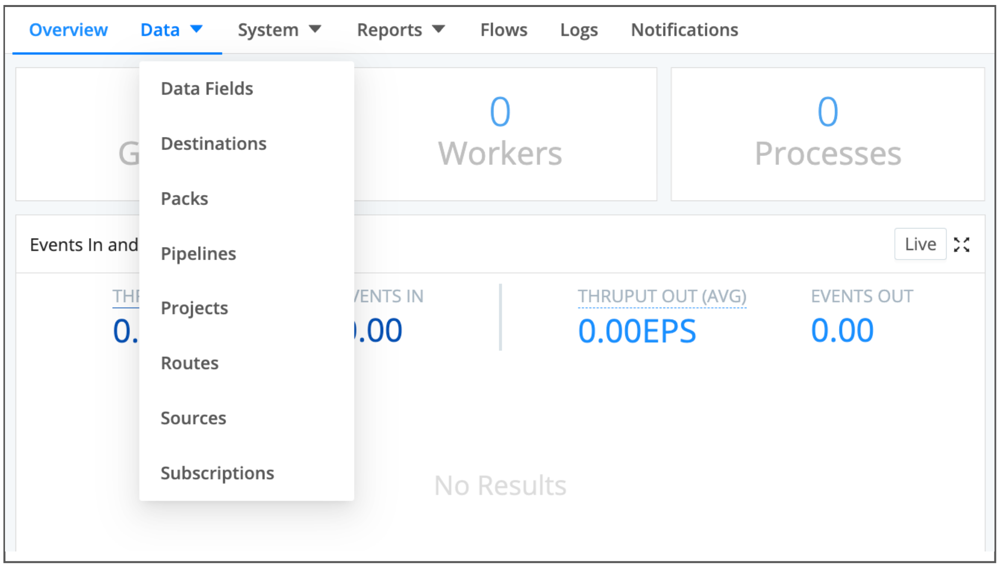
Monitoring page

Except for these configuration change markers, Monitoring data does not persist across Cribl Stream restarts. Keep this in mind before you restart the server.

Data Monitoring

From the **Monitoring** page, open the **Data** submenu to isolate throughput for any of the following:

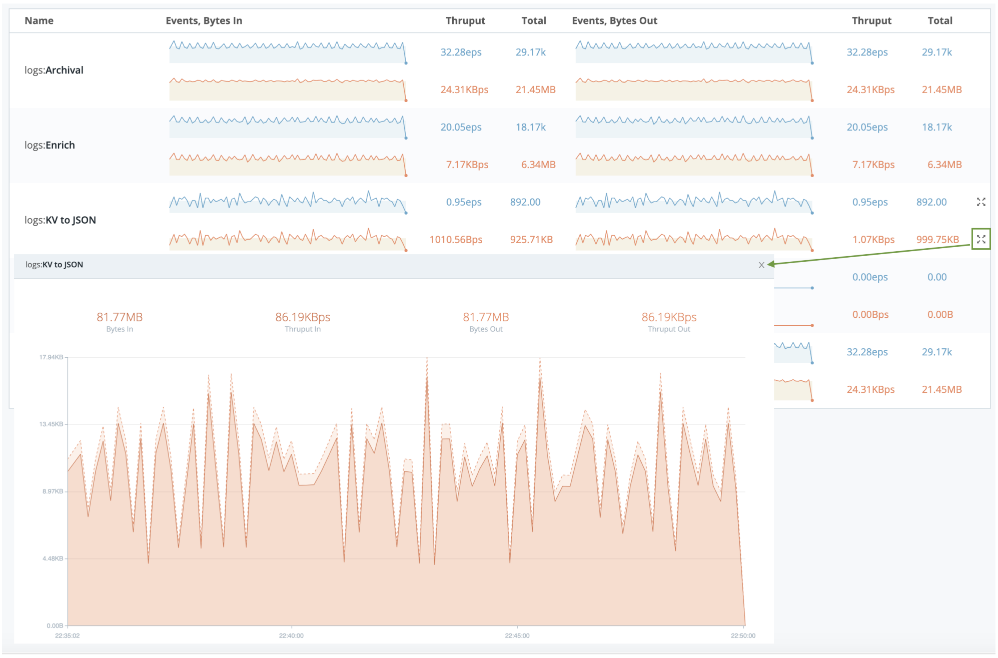
- Data Fields
- Destinations
- Packs
- Pipelines
- Projects
- Routes
- Sources
- Subscriptions



Monitoring > Data submenu

Sparklines

Dense displays are condensed to sparklines for legibility. Hover over the right edge to display Maximize buttons that you can click to zoom these up to detailed graphs.



Name	Events, Bytes In	Thruput	Total	Events, Bytes Out	Thruput	Total
logs:**Archival**		32.28eps	29.17k		32.28eps	29.17k
		24.31KBps	21.45MB		24.31KBps	21.45MB
logs:**Enrich**		20.05eps	18.17k		20.05eps	18.17k
		7.17KBps	6.34MB		7.17KBps	6.34MB
logs:**KV to JSON**		0.95eps	892.00		0.95eps	892.00
		1010.56Bps	925.71KB		1.07KBps	999.75KB
					0.00eps	0.00
					0.00Bps	0.00B
					32.28eps	29.17k
					24.31KBps	21.45MB

logs:**KV to JSON**

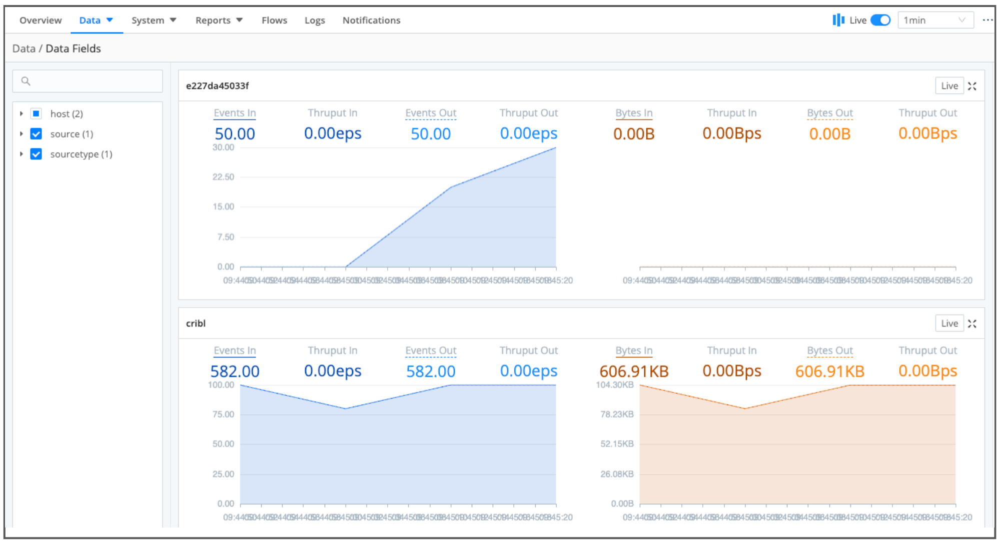
Sparklines and fly-out

Data Fields

The **Data Fields** page lets you preview the flow of events that contain a specific data field. The left (blue) side summarizes events in/out. The right (orange) side summarizes bytes in/out.

You can control which data fields are included in these graphs with the Disable field metrics setting.

To add data fields to this setting, navigate to **Settings**, then select **Global Settings**. From here, select **General Settings**, then **Metrics** (under **Limits**).



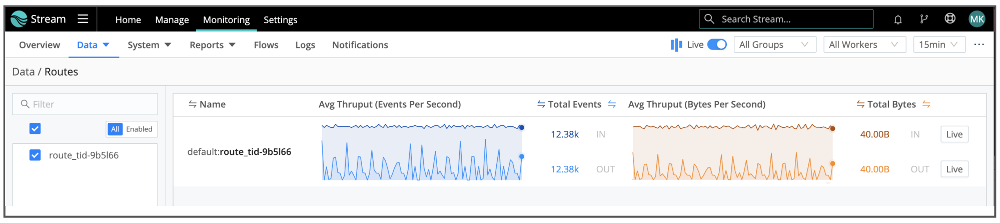
Data Fields graph in Monitoring

Disable field metrics settings are not available on Cribl.Cloud-managed Worker Nodes.

Routes

The **Routes** page condenses multiple details and options:

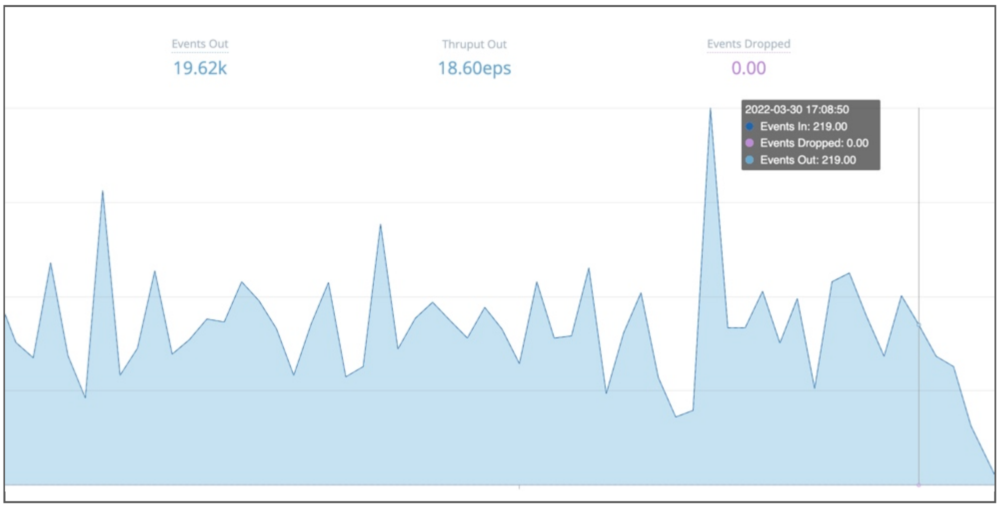
- Each row independently summarizes throughput for a separate Route.
- The left (blue) side summarizes events in/out. The right (orange) side summarizes bytes in/out.
- On each row, the top number summarizes average events/bytes into the Route, and the bottom number summarizes events/bytes out. Both are averaged over your selected time granularity.
- On each row, the upper Maximize button zooms up the left (events) graph, and the lower Maximize button zooms up the right (bytes) graph.



Routes page’s multiple rows and controls

Fly-Out Details

You can hover over graphs to display a fly-out with details. On the **Free Memory** and **CPU Load Average** graphs, fly-outs are limited to a manageable 10 Worker Nodes, even if more Nodes are active. These fly-outs will occasionally show transient `0` metrics for some Workers, because Cribl Stream prioritizes reporting current throughput over memory/load metrics.

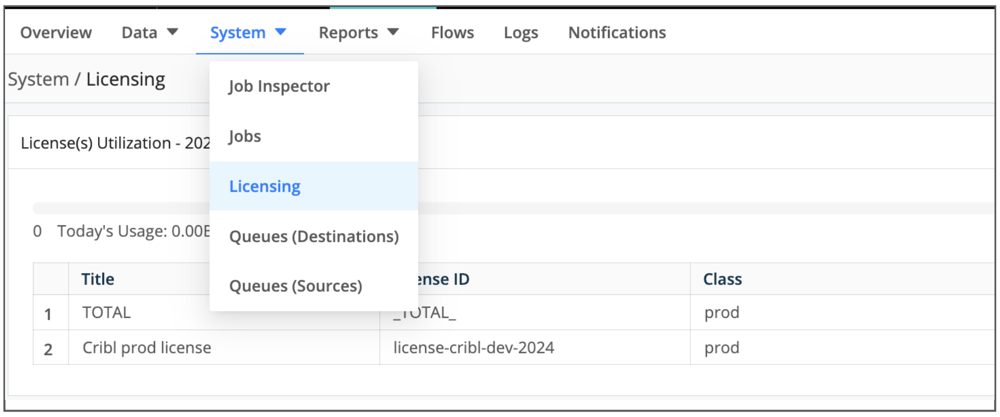


Throughput details

System Monitoring

From the **Monitoring** page, open the **System** submenu to isolate throughput for any of the following:

- Job Inspector
- Jobs (and tasks in-flight, see Collector Sources)
- Leaders (available only with High Availability enabled; see Second Leader)
- Licensing
- Queues (Destinations)
- Queues (Sources)



License(s) Utilization - 202

0 Today's Usage: 0.00E

	Title	ense ID	Class
1	TOTAL	_TOTAL_	prod
2	Cribl prod license	license-cribl-dev-2024	prod

Monitoring > System submenu

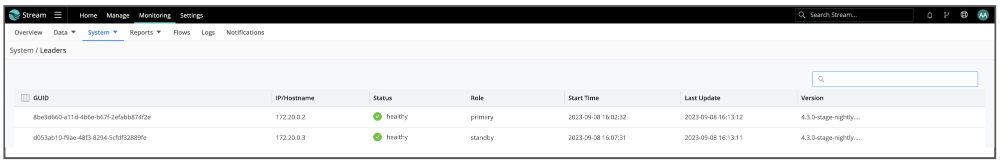
Job Inspector

Select **System** then **Job Inspector** from the Monitoring page to view and manage pending, in-flight, and completed collection jobs and their tasks. For details about the resulting page, see Monitoring and Inspecting Collection Jobs.

Leaders

This menu item is available only in distributed deployments with High Availability (HA) enabled.

Select **Monitoring**, then **System** and **Leaders** to view the status of your Leader Nodes. For more information on how to configure a second Leader Node for failover/durability, see Second Leader).



System / Leaders

GUID	IP/Hostname	Status	Role	Start Time	Last Update	Version
8be3d660-a11d-4b6e-b67f-2efabb874f2e	172.20.0.2	✓ healthy	primary	2023-09-08 16:02:32	2023-09-08 16:13:12	4.3.0-stage-nightly....
d053ab10-f9ae-48f3-8294-5cfdf32889fe	172.20.0.3	✓ healthy	standby	2023-09-08 16:07:31	2023-09-08 16:13:11	4.3.0-stage-nightly....

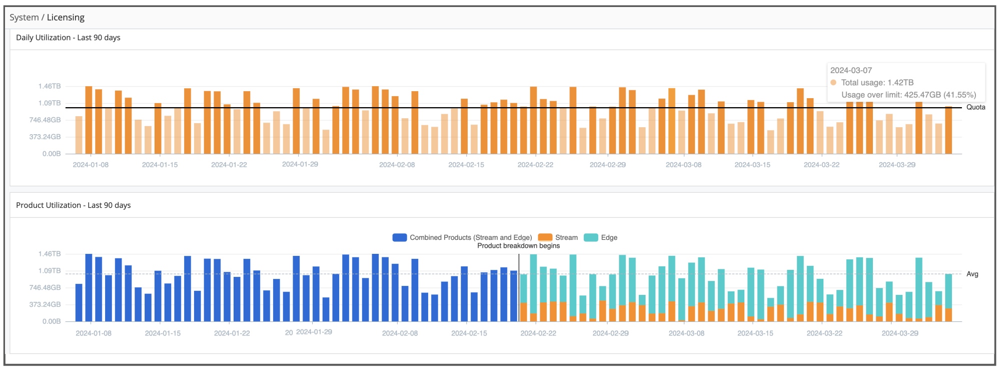
Monitoring Leader Nodes

Licensing

Select **System**, then **Licensing** from the Monitoring page to check the expiration dates of your licenses, daily data throughput, and events quotas. You can also compare your daily Stream and Edge data throughput against your license quota – and against granular and average throughput over the last 30 to 365 days.

In this graph, you'll see the following:

- A horizontal bar that indicates your quota limit.
- Combined legacy data represented by blue bars that show licensing usage prior to v4.6.0, when Edge data and Stream data was separated.
- Tooltips display details about data usage, data amount over/under license quota, Edge dropped bytes, and data percentage over/under license quota.
- Dots on the daily usage bar graph represent configuration changes in the system.
- The **Daily Events In** chart only shows events that count towards your license, broken out by product (Stream and Edge). It filters out `datagen`, `cribl_http`, and `cribl_tcp` events.



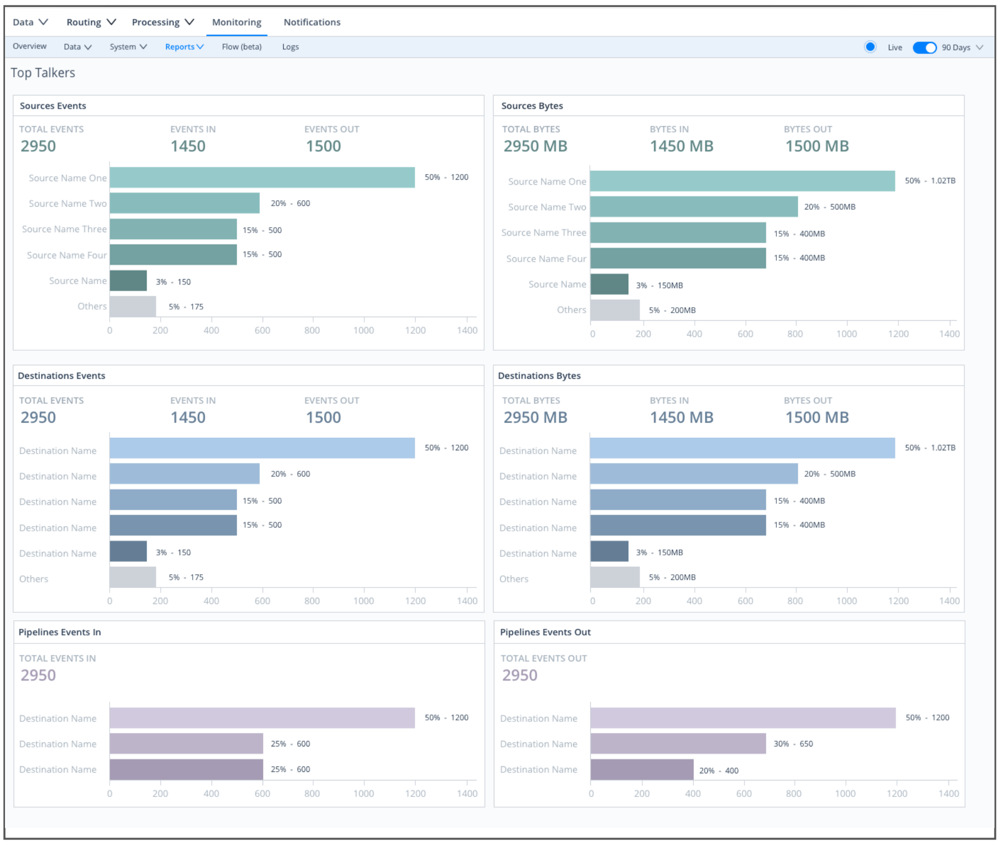
Licensing breakdown from the Monitoring page

Even on single-instance deployments, you must have `git` installed in order for the Monitoring > **Licensing** page to display configuration change markers.

For the most current and accurate throughput data, enable the Cribl Internal Source's CriblMetrics option. Forward metrics to your metrics Destination of choice, and run a report on `cribl.total.in_bytes`. CriblMetrics aggregates metrics at the Worker Process level, every 2 seconds.

Reports/Top Talkers

Select **Monitoring** > **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.



Monitoring > Reports/Top Talkers

Flows

Select **Flows** from the Monitoring page's top nav or ••• overflow menu to see a graphical, left-to-right visualization of data flow through your Cribl Stream deployment.

Internal Logs and Metrics

Select **Logs** from the Monitoring page's top nav. Cribl Stream'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. (Details below.)

Types of Logs

Cribl Stream 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: `$CRIBL_HOME/log/cribl.log`
- Worker Process(es) Logs – These logs are emitted by all the Worker Processes, and are very common on single-instance deployments and Workers. Filesystem location: `$CRIBL_HOME/log/worker/N/cribl.log`
- Worker Group/Fleet Logs – These logs are emitted by all processes that help a Leader Node configure Worker Groups/Fleets. Filesystem location: `$CRIBL_HOME/log/group/GROUPNAME/cribl.log`

In a distributed deployment (Stream), all Workers forward their metrics to the Leader Node, which then consolidates them to provide a deployment-wide view.

For details about generated log files, see Internal Logs. To work with logs in Cribl Stream's UI, see Search Internal Logs.

Log Rotation and Retention

Cribl Stream rotates logs every 5 MB, keeping the most recent 5 logs. Logs will reach the 5 MB threshold more quickly with verbose logging levels (such as `debug`) and with high volumes of system activity.

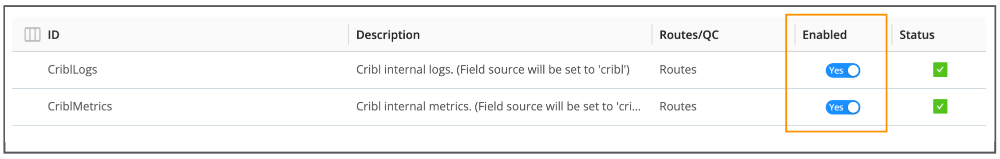
For long-term retention, Cribl recommends sending logs from the Leader, and from Workers' Cribl Internal > CriblLogs Sources, to a downstream service of your choice. See the next section for details.

Forward Logs and Metrics Externally

Cribl Stream supports forwarding internal logs and metrics to your preferred external monitoring solution. The Cribl Internal Source captures internal logs and metrics to send to Destinations.

In the following steps, we'll use the graphical QuickConnect UI to set up the **Cribl Internal** Source and how data is sent to your Destination. See Cribl Internal for details on how to instead configure the **Cribl Internal** Source via the **Data** > **Sources** (Stream) or **More** > **Sources** (Edge) submenus.

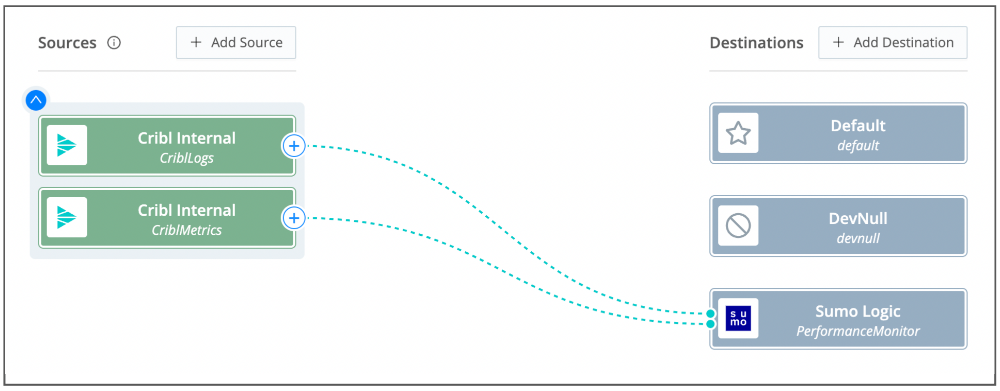
1. From the top nav, click **Manage**, then select a **Worker Group** to configure.
2. To open **QuickConnect**: Stream – select the **QuickConnect** tile on the **Overview** page. Edge – click the **Collect** submenu.
3. Click **Add Source**.
4. From the resulting drawer's tiles, select **System and Internal**, then hover over the **Cribl Internal** tile.
5. Click **Select Existing**.
6. On the **CriblLogs** and/or the **CriblMetrics** row, toggle **Enabled** to `Yes`. Confirm your choice in the resulting message box.



ID	Description	Routes/QC	Enabled	Status
CriblLogs	Cribl internal logs. (Field source will be set to 'cribl')	Routes	Yes	✓
CriblMetrics	Cribl internal metrics. (Field source will be set to 'cri...	Routes	Yes	✓

Cribl Internal Sources – click to configure

1. Notice how the **Routes/QC** column still says **Routes**. We want to use QuickConnect, so we'll change this by clicking on the **CriblLogs** and/or the **CriblMetrics** row again. Once clicked, you'll confirm **Yes** to the message box asking to switch the Source to send to QuickConnect instead of Routes.
2. Click and drag the **+** button on the right side of the Source to your desired Destination. A **Connection Configuration** modal will prompt you to select a **Passthru**, **Pipeline**, or **Pack connection**. See QuickConnect for details.



Cribl Internal forwarding to Sumo Logic

1. This will send internal logs and metrics to your Destination, 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 Worker Process logs (see Types of Logs above). You can, however, use a Script Collector to listen for API Server or Worker Group events.

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

Controlling Metrics Volume

To reduce the volume of metrics sent through Cribl Stream, see options on the Cribl Internal Source.

Disable Field Metrics

To send fewer metrics to the Leader Node, you can specify these metrics types in the blocklist at **Settings** > (**Global Settings**) > **System** > **General Settings** > **Limits** > **Metrics** > **Disable field metrics**.

The default fields to ignore are `host`, `source`, `sourcetype`, `index`, and `project`. You can remove any of the defaults, or add other fields you do not want to send as metrics.

However, when you enable the **Cribl Internal** Source, Cribl Stream ignores this **Disable field metrics** setting, and full-fidelity data will flow down the Routes.

Dropping Metrics

When the number of in-flight requests for sending metrics from Worker to Leader exceeds a limit (1,000 requests), Workers will stop sending metrics. Dropped metrics information is logged under `channel="clustercomm"`.

You can exclude certain metrics from being dropped due to exceeding limits. Specify these metrics at **Settings** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics never-drop list**.

This setting is available only on Worker Nodes.

Metrics Garbage Collection

Metrics garbage collection (GC) runs when the total number of stored metrics exceeds the max metrics limit (default 1,000,000). Metrics are then removed, starting with the oldest ones. This happens both on Workers and Leaders.

You can define how often to run garbage collection on each Worker Group. Select **Group Settings** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics GC period**. The default is 60 seconds.

Metrics Tracking by Worker ID

Typically, all metrics are assigned their own Worker Node ID dimensions so they can be split by worker if needed.

You can define which metrics are **not** assigned a Worker Node ID dimension by adding them to the list at **Group Settings** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics worker tracking**.

Control Metrics Lag and Disk Usage

If one or more Worker Groups has a large number of enabled Sources, and clicking into these Sources does not promptly display their status, a workaround is to prevent Cribl Stream from writing metrics to disk.
On the Leader (versions 4.1.2 and later), navigate to **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Storage**, and disable the **Persist metrics** toggle. Then restart the Leader.

If you keep **Persist metrics** enabled, you can use the adjacent **Metrics max disk space** field to control the written metrics' footprint. This threshold defaults to `64 GB`.

Cardinality Limit

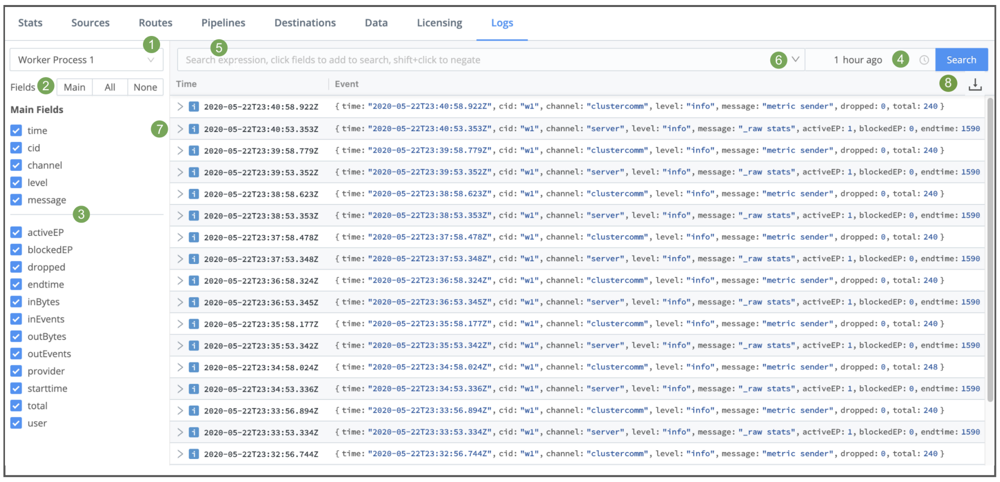
You can define the cardinality limit or metrics on each Worker Group. Select **Group Settings** > **System** > **General Settings** > **Limits** > **Metrics** > **Metrics cardinality limit** The default is 1,000.

Refer to **Monitoring** > **Data** > **Data Fields** to identify the fields that have the highest cardinality.

Search Internal Logs

Cribl Stream exists because logs are great and wonderful things! Using Cribl Stream's **Monitoring > Logs** page, you can search all Cribl Stream's internal logs at once – from a single location, for both Leader and Worker. This enables you to query across all internal logs for strings of interest.

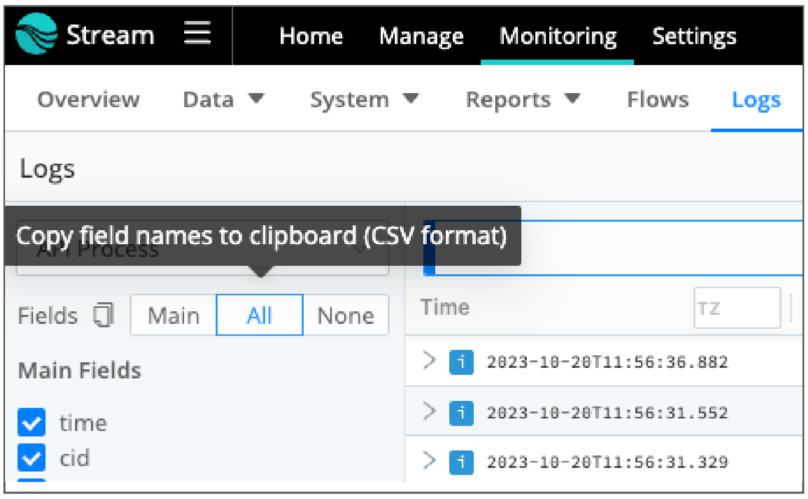
The labels on this screenshot highlight the key controls you can use (see the descriptions below):



```
2020-05-22T23:40:58.922Z  { time: "2020-05-22T23:40:58.922Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:40:53.353Z  { time: "2020-05-22T23:40:53.353Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:39:58.779Z  { time: "2020-05-22T23:39:58.779Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:39:53.352Z  { time: "2020-05-22T23:39:53.352Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:38:58.623Z  { time: "2020-05-22T23:38:58.623Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:38:53.353Z  { time: "2020-05-22T23:38:53.353Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:37:58.478Z  { time: "2020-05-22T23:37:58.478Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:37:53.348Z  { time: "2020-05-22T23:37:53.348Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:36:58.324Z  { time: "2020-05-22T23:36:58.324Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:36:53.345Z  { time: "2020-05-22T23:36:53.345Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:35:58.177Z  { time: "2020-05-22T23:35:58.177Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:35:53.342Z  { time: "2020-05-22T23:35:53.342Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:34:58.024Z  { time: "2020-05-22T23:34:58.024Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 248 }
2020-05-22T23:34:53.336Z  { time: "2020-05-22T23:34:53.336Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:33:56.894Z  { time: "2020-05-22T23:33:56.894Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:33:53.334Z  { time: "2020-05-22T23:33:53.334Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 1, blockedEP: 0, endtime: 1590
2020-05-22T23:32:56.744Z  { time: "2020-05-22T23:32:56.744Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
```

Logs page (controls highlighted)

1. **Log file selector**: Choose the Node to view. In a distributed deployment (Stream), this list will be hierarchical, with Workers displayed inside their Leader.
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.

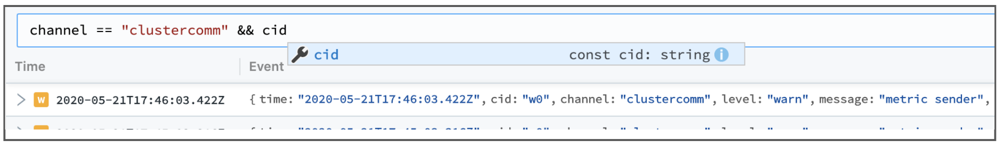


```
2023-10-20T11:56:36.882
2023-10-20T11:56:31.552
2023-10-20T11:56:31.329
```

Monitoring - Copy Fields Icon

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.

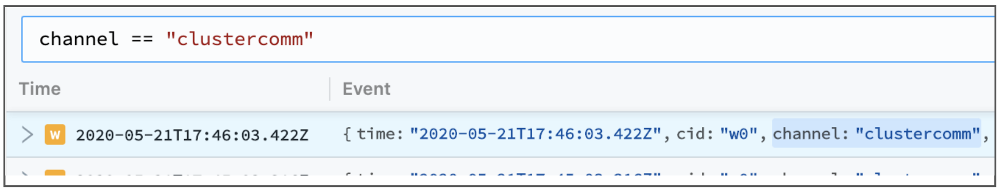
Typeahead assist is available for expression completion:



```
channel == "clustercomm" && cid
cid                                   const cid: string
Time            Event
2020-05-21T17:46:03.422Z   { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm", level: "warn", message: "metric sender",
```

Expression autocompletion

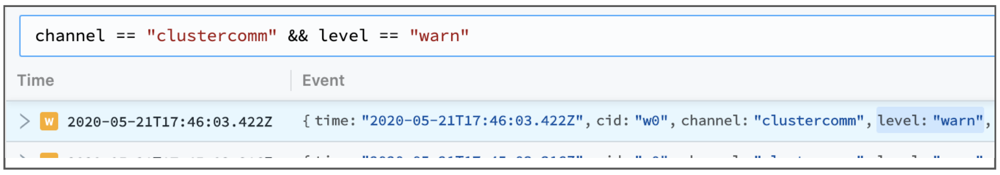
Click a field in any event to add it to a query:



```
channel == "clustercomm"
Time            Event
2020-05-21T17:46:03.422Z   { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm",
```

Click to add a field

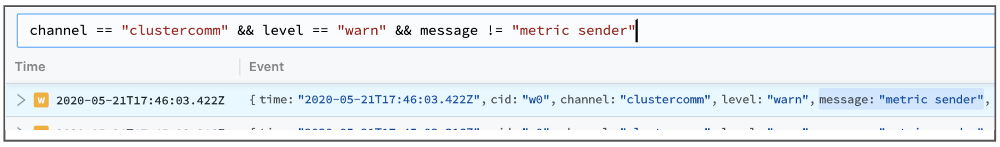
Click other fields to append them to a query:



```
channel == "clustercomm" && level == "warn"
Time            Event
2020-05-21T17:46:03.422Z   { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm", level: "warn",
```

Click to append a field

Shift+click to *negate* a field:

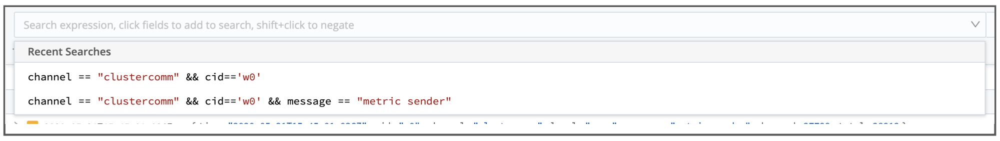


```
channel == "clustercomm" && level == "warn" && message != "metric sender"
Time            Event
2020-05-21T17:46:03.422Z   { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm", level: "warn", message: "metric sender",
```

Shift-click to negate a field

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:

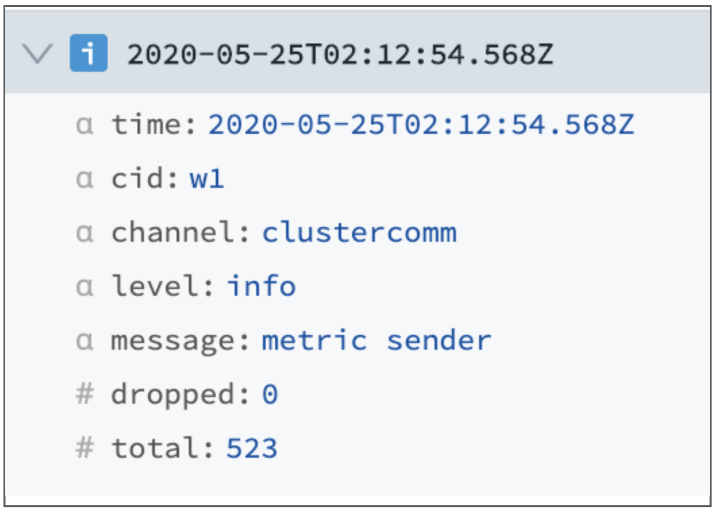


```
Search expression, click fields to add to search, shift+click to negate
Recent Searches
channel == "clustercomm" && cid=='w0'
channel == "clustercomm" && cid=='w0' && message == "metric sender"
```

Retrieve recent searches

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:



```
2020-05-25T02:12:54.568Z
α time: 2020-05-25T02:12:54.568Z
α cid: w1
α channel: clustercomm
α level: info
α message: metric sender
# dropped: 0
# total: 523
```

Log event details

8. **Export Logs as JSON** button: Exports logs as a file initially named `CriblMonitoringLogs.json`. (You can edit this name upon export.) The logs' scope will be filtered both by your Time range selector setting, and by how fully you've scrolled down to lazy-load that time range into the displayed UI.

Logging Settings

On Cribl Stream'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 per Worker Group.

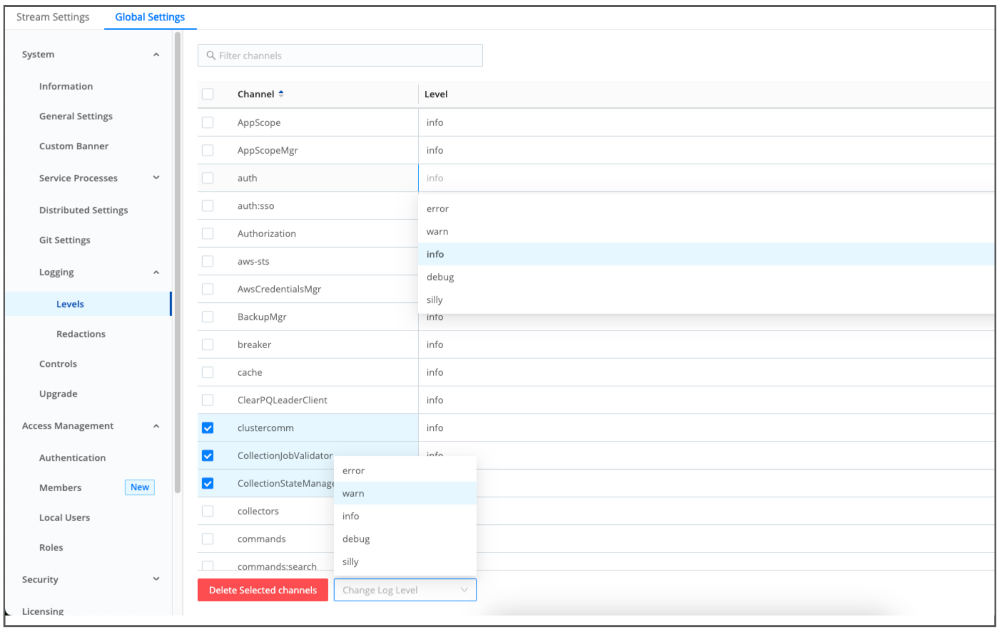
Change Logging Levels

To adjust logging levels:

- In a single-instance deployment, select **Settings** > **System > Logging > Levels**.
- For the Leader Node's own logs, select **Settings** > **Global Settings** > **System > Logging > Levels**.
- In a distributed deployment, first select **Manage**, then click into the group you want to configure. Next, select **Group Settings** (or **Fleet 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.



	Channel	Level
☐	AppScope	info
☐	AppScopeMgr	info
☐	auth	info
☐	auth:sso	
☐	Authorization	
☐	aws-sts	
☐	AwsCredentialsMgr	
☐	BackupMgr	info
☐	breaker	info
☐	cache	info
☐	ClearPQLeaderClient	info
✓	clustercomm	info
✓	CollectionJobValidator	info
✓	CollectionStateManage	
☐	collectors	
☐	commands	
☐	commands:search	

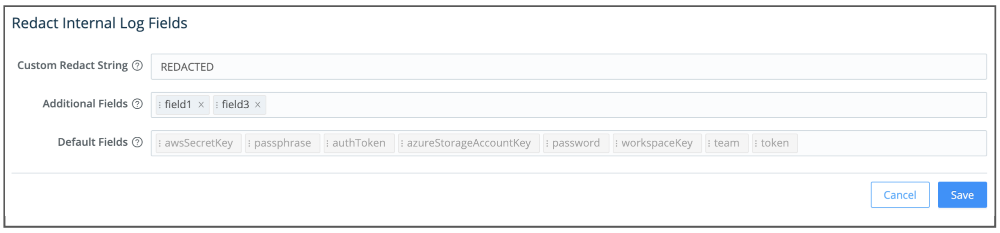
Manage Logging Levels page

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 Stream's internal logs. To access these settings:

- In a single-instance deployment, select **Settings** > **System** > **Logging** > **Redactions**.
- In a single-instance deployment, or for the Leader Node's own logs, select **Settings** > **Global Settings** > **System** > **Logging** > **Redactions**.
- In a distributed deployment, first select **Manage**, then click into the group you want to configure. Next, select that group's **Group Settings** > **System** > **Logging** > **Redactions**.


Redact Internal Log Fields

Custom Redact String	REDACTED
Additional Fields	field1, field3
Default Fields	awsSecretKey, passphrase, authToken, azureStorageAccountKey, password, workspaceKey, team, token

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 Stream 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 Stream's default redaction pattern (explained below) on the affected fields.

Default Redact String

By default, Cribl Stream 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.

Health Endpoint

Each Cribl Stream 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 Stream 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, `9000`).

12.1. Internal Metrics

When sending Cribl Stream 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 Stream reports its internal metrics within the Cribl Stream UI, in the same way that it reports internal logs. Navigate to **Monitoring** > **Logs** (Cribl Stream), or to a Fleet's **Health** > **Sources** tab (Cribl Edge). To expose metrics for capture or routing, enable the **Cribl Internal** Source > **CriblMetrics** section. In Cribl Stream 4.4 and later, you can export metrics (and logs) to JSON files.

By default, Cribl Stream 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 Stream do not monotonically increase or decrease. They are reset at the end of each reporting period. Cribl Stream 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 Stream 4.1.2. See Dimensions for their replacements.
- Disabled Sources no longer generate Cribl internal metrics as of Cribl Stream 4.2.

Total Throughput Metrics

Five important metrics below are prefixed with `total.` These power the top of Cribl Stream'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 Stream'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 Stream 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. On the Destination side, use the `#output=<id>` dimension. For example, `pq.out_events|#output=kafka` would be one unique metric; `pq.out_events|#output=camus` would be another.

Other Internal Metrics

The Cribl Internal Source emits other metrics that, when displayed in downstream dashboards, can be useful for understanding Cribl Stream's behavior and health. These include:

- `cribl.logstream.total.activeCxn` – Total active inbound TCP connections.
- `cribl.logstream.pipe.in_events` – Inbound events per Pipeline.
- `cribl.logstream.pipe.out_events` – Outbound events per Pipeline.
- `cribl.logstream.pipe.dropped_events` – Dropped events per Pipeline.
- `cribl.logstream.metrics_pool.num_metrics` – The total number of unique metrics that have been allocated into memory.
- `cribl.logstream.collector_cache.size` – Each Collector function (`default/cribl/collectors/<collector>/index.js`) is loaded/initialized only once per job, and then cached. This metric represents the current size of this cache.
- `cribl.logstream.cluster.metrics.sender.inflight` – Number of metric packets currently being sent from a Worker/Edge Node Process to the API Process, via IPC (interprocess communication).
- `cribl.logstream.backpressure.outputs` – Destinations experiencing backpressure, causing events to be either blocked or dropped.
- `cribl.logstream.blocked.outputs` – Blocked Destinations. (This metric is more restrictive than the one listed just above.)
- `cribl.logstream.pq.queue_size` – Current queue size, in bytes, per Worker Process.
- `cribl.logstream.host.in_bytes` – Inbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.in_events` – Inbound events from a given host (host is a characteristic of the data).
- `cribl.logstream.host.out_bytes` – Outbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.out_events` – Outbound events from a given host (host is a characteristic of the data).
- `cribl.logstream.index.in_bytes` – Inbound bytes per index.
- `cribl.logstream.index.in_events` – Inbound events per index.

- `cribl.logstream.index.out_bytes` – Outbound bytes per index.
- `cribl.logstream.index.out_events` – Outbound events per index.
- `cribl.logstream.route.in_bytes` – Inbound bytes per Route.
- `cribl.logstream.route.in_events` – Inbound events per Route.
- `cribl.logstream.route.out_bytes` – Outbound bytes per Route.
- `cribl.logstream.route.out_events` – Outbound events per Route.
- `cribl.logstream.source.in_bytes` – Inbound bytes per Source.
- `cribl.logstream.source.in_events` – Inbound events per Source.
- `cribl.logstream.source.out_bytes` – Outbound bytes per Source.
- `cribl.logstream.source.out_events` – Outbound events per Source.
- `cribl.logstream.sourcetype.in_bytes` – Inbound bytes per sourcetype.
- `cribl.logstream.sourcetype.in_events` – Inbound events per sourcetype.
- `cribl.logstream.sourcetype.out_bytes` – Outbound bytes per sourcetype.
- `cribl.logstream.sourcetype.out_events` – Outbound events per sourcetype.
- `nodecluster.primary.messages` – Number of IPC (interprocess communication) calls made by a Worker/Edge Node Process to the API Process.

Dimensions

Cribl Stream 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 Worker Group 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 Stream 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.

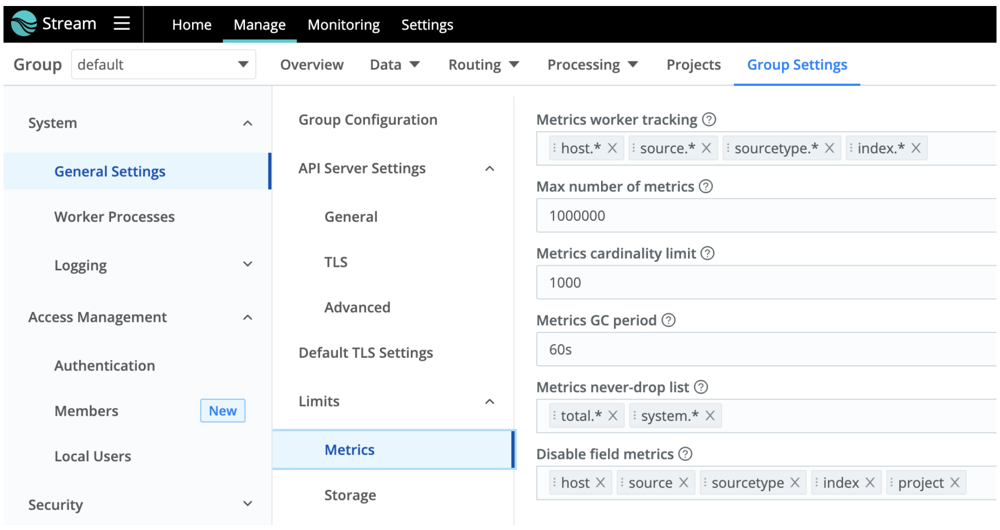
Controlling Metrics

You can control the volume of internal metrics that Cribl Stream emits, by adjusting Metrics Limits. You access these as follows, per deployment type:

- Single-instance: **Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Distributed Leader: **Settings** > **Global Settings** > **System** > **General Settings** > **Limits** > **Metrics**.
- Distributed Worker Groups: [Select a Worker Group] > **Group Settings** > **System** > **General Settings** > **Limits** > **Metrics**.

Here, you can adjust the maximum number of metric series allowed, metrics' cardinality, metrics to always include (**Metrics never-drop list**), metrics to always exclude in order to optimize performance (**Disable field metrics**), and other options.

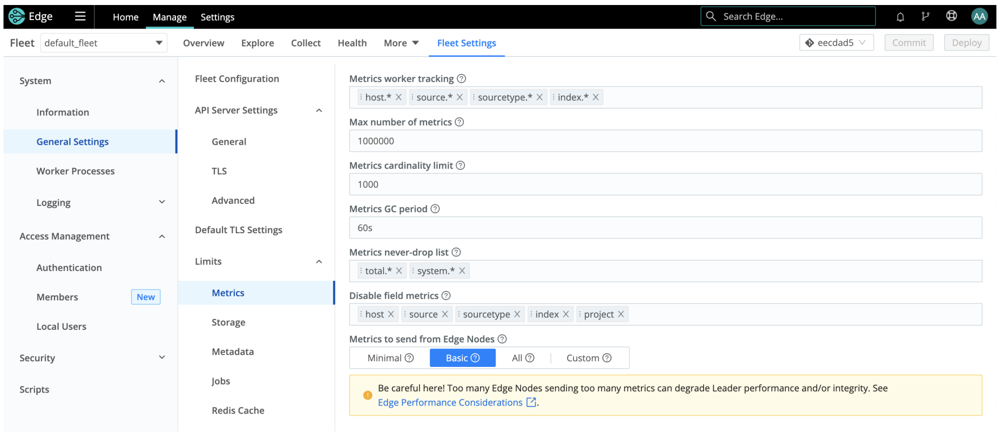
Here's how these controls look in Cribl Stream:



Metrics worker tracking: host.* source.* sourcetype.* index.*
Max number of metrics: 1000000
Metrics cardinality limit: 1000
Metrics GC period: 60s
Metrics never-drop list: total.* system.*
Disable field metrics: host source sourcetype index project

Metrics settings for Cribl Stream Worker Groups

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



Metrics worker tracking: host.*, source.*, sourcetype.*, index.*

Max number of metrics: 1000000

Metrics cardinality limit: 1000

Metrics GC period: 60s

Metrics never-drop list: total.*, system.*

Disable field metrics: host, source, sourcetype, index, project

Metrics to send from Edge Nodes: Minimal [✓ Basic] All Custom

Be careful here! Too many Edge Nodes sending too many metrics can degrade Leader performance and/or integrity. See Edge Performance Considerations.

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`,

Set	Purpose	List of Metrics Sent

		`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 Stream'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 Stream 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.

12.2. Internal Logs

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

You can display and export all internal logs from the UI at **Monitoring** > **Logs** (Stream) or **Manage** > **Logs** (Edge). Logs' persistence depends on event volume, not time – for details, see Log Rotation and Retention.

> Currently, most logs listed on this page are available only in customer-managed deployments. Cribl.Cloud currently supports only Worker Node logs, and only on hybrid (not Cribl-managed) Workers.

Leader Node Logs (Distributed)

The `API/main` process emits the following logs into the Leader Node's `$CRIBL_HOME/log/` directory.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Principal log in Cribl Stream. Includes telemetry/license-validation logs. Corresponds to top-level `cribl.log` on Diag page.	**Leader** > **API Process**
`access.log`	API calls, e.g., `GET /api/v1/version/info`.	**Leader** > **Access**
`audit.log`	Actions pertaining to files, e.g., `create`, `update`, `commit`, `deploy`, `delete`.	**Leader** > **Audit**
`notifications.log`	Messages that appear in the notification list in the UI.	**Leader** > **Notifications**
`ui-access.log`	Interactions with different UI components described as URLs, e.g., `/settings/apidocs`, `/dashboard/logs`.	**Leader** > **UI Access**

The `API/main` process emits the following service logs into the Leader Node's `$CRIBL_HOME/log/service/` directory. Each service includes a `cribl.log` file that logs the service's internal telemetry and an `access.log` file that logs which API calls the service has handled.

SERVICE NAME	DESCRIPTION	EQUIVALENT ON Logs PAGE
Connections Service	Handles all worker connections and communication, including heartbeats, bundle deploys, teleporting, restarting, etc. Workers are assigned to connection processes using a round-robin algorithm.	**Leader** > **Connections Service**
Lease Renewal Service	Handles lease renewal for the primary Leader Node.	**Leader** > **Lease Renewal Service**
Metrics Service	Handles in-memory metrics, merging of incoming packets, metrics persistence and rehydration, and UI queries for metrics.	**Leader** > **Metrics Service**
Notifications Service	Triggers notifications based on its configuration.	**Leader** > **Notifications Service**

The Config Helper process for each Worker Group/Fleet emits the following log in `$CRIBL_HOME/log/group/GROUPNAME`.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about config maintenance, previews, etc.	**GROUPNAME** > **Config helper**

Worker Node Logs (Distributed)

The API Process emits the following log in `$CRIBL_HOME/log/`.

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node communicating with the Leader Node (i.e., with its API Process) and other API requests, e.g., sending metrics, reaping job artifacts.	**GROUPNAME** > **Worker:HOSTNAME** > **API Process**

Each Worker Process emits the following logs in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID. (The `metrics.log` file is written only when HTTP-based Destinations exist.)

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node processing data.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

Logfile Name	Description	Equivalent on Logs page
`metrics.log`	Messages about the Worker/Edge Node's outbound HTTP request statistics.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

For convenience, the UI aggregates the Worker/Edge Node Process logs as follows.

Logfile Name	Description	Equivalent on Logs page
N/A	Aggregation of all the `Worker Process N` logs and the API Process log.	**GROUPNAME** > **WORKER_NAME**

The Worker Node logs listed above are currently available only on hybrid (not Cribl-managed) Workers. The single-instance logs listed below are not relevant to Cribl.Cloud.

Single-Instance Logs

The `API/main` process emits the same logs as it does for a distributed deployment, in `$CRIBL_HOME/log/`:

- `cribl.log`
- `access.log`
- `audit.log`
- `notifications.log`
- `ui-access.log`

Each Worker/Edge Node Process emits the following logs in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID. (The `metrics.log` file is written only when HTTP-based Destinations exist.)

Logfile Name	Description	Equivalent on Logs page
`cribl.log`	Messages about the Worker/Edge Node processing data.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**
`metrics.log`	Messages about the Worker/Edge Node's outbound HTTP request statistics.	**GROUPNAME** > **Worker:HOSTNAME** > **Worker Process N**

`_raw stats` Event Fields

Each Worker/Edge Node Process logs this information at a 1-minute frequency. So each event's scope covers only that Worker/Edge Node Process, over a 1-minute time span defined by the `startTime` and `endTime` fields.

Sample Event

```
{"time":"2022-11-17T16:54:05.349Z","cid":"w0","channel":"server","level":"info","me
```

Field Descriptions

Field	Description
`abortCxn`	Number of TCP connections that were aborted.
`activeCxn`	Number of TCP connections newly opened at the time the `_raw` stats are logged. (This is a gauge when exported in internal metrics, and can otherwise be ignored as an instantaneous measurement. Only some application protocols count toward this; e.g., any HTTP-based Source does not count.)
`activeEP`	Number of currently active event processors (EPs). EPs are used to process events through Breakers and Pipelines as the events are received from Sources and sent to destinations. EPs are typically created per TCP connection (such as for HTTP).
`blockedEP`	Number of currently blocked event processors (caused by blocking Destinations).
`closeCxn`	Number of TCP connections that were closed.
`cpuPerc`	CPU utilization from the combined user and system activity over the last 60 seconds.
`droppedEvents`	This is equivalent to the `total.dropped_events` metric. Drops can occur from Pipeline Functions, from Destination Backpressure, or from other errors. Any event not sent to a Destination is considered dropped.
`eluPerc`	Event loop utilization. Represents the percentage of time over the last 60 seconds that the NodeJS runtime spent processing events within its event loop.
`endTime`	The end of the timespan represented by these metrics. (Will always be 60 seconds after `startTime`.)
`inBytes`	Number of bytes received from all Sources (based only off `_raw`).
`inEvents`	Number of events received from all inputs after Event Breakers are applied. This can be larger than `outEvents` if events are dropped via Drop, Aggregation, Suppression, Sampling, or similar Functions.

Field	Description
`mem.buffers`	Memory allocated for ArrayBuffers and SharedArrayBuffers.
`mem.ext`	External section of process memory, in MB.
`mem.heap`	Used heap section of process memory, in MB.
`mem.heapTotal`	Total heap section of process memory, in MB.
`mem.rss`	Resident set size section of process memory, in MB.
`openCxn`	Same as `activeCxn`, but tracked as a counter rather than a gauge. So `openCxn` will show all connections newly opened each minute, and is more accurate than using `activeCxn`.
`outBytes`	Number of bytes sent to all Destinations (based only off `_raw`).
`outEvents`	Number of events sent out to all Destinations. This can be larger than `inEvents` due to creating event clones or entirely new unique events (such as when using the Aggregation Function).
`pqInBytes`	Number of bytes that were written to Persistent Queues, across all Destinations.
`pqInEvents`	Number of events that were written to Persistent Queues, across all Destinations.
`pqOutBytes`	Number of bytes that were flushed from Persistent Queues, across all Destinations.
`pqOutEvents`	Number of events that were flushed from Persistent Queues, across all Destinations.
`pqTotalBytes`	Amount of data currently stored in Persistent Queues, across all Destinations.
`rejectCxn`	Number of TCP connections that were rejected.
`startTime`	The beginning of the timespan represented by these metrics.
`tasksCompleted`	The number of tasks the process has started and completed for all collection jobs for which it was executing tasks.
`tasksStarted`	The number of tasks the process started for all collection jobs for which it was executing tasks.

13. Working With Data

13.1. Event Model

All data processing in Cribl Stream 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 Stream 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:

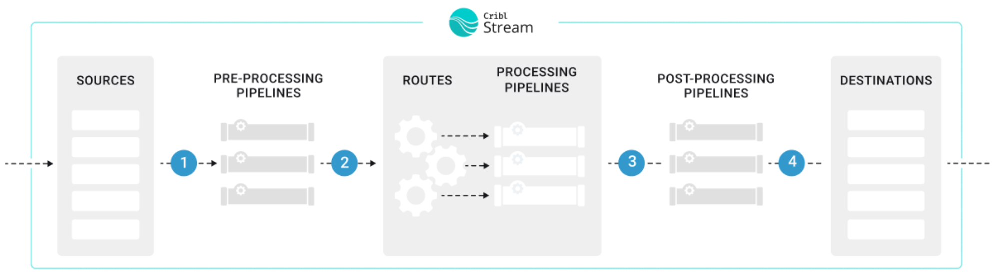
- Fields that start with a double-underscore are known as **internal fields**, and each Source can add one or many to each event. For example, Syslog adds both a `__inputId` and a `__srcIpPort` field. Internal fields are used only within Cribl Stream, and are not passed down to Destinations.
- Upon arrival 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`. (When Cribl Stream emits events based on incoming Kafka messages, it handles the `_time` field a bit differently.)
- 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):

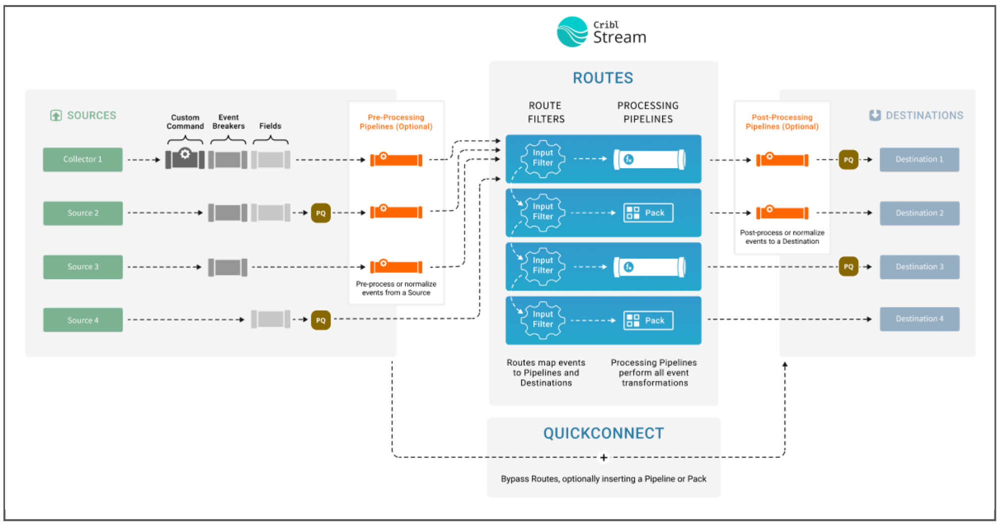
1. Click **Start a Capture**.
2. In the resulting modal, enter a **Filter expression** to narrow down the events of interest.
3. Click **Capture...** and (optionally) change the default Time and/or Event limits.
4. Select the desired **Where to capture** option. There are four options:

- **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 any Routes. (QuickConnect bypasses Routes, and can bypass processing Pipelines.)
- **3. Before the post-processing Pipeline** – Capture events right after any Processing Pipeline that handled them, before any post-processing Pipeline.
- **4. Before the Destination** – Capture events right after any post-processing Pipeline, before they go out to the configured Destination.



13.2. Event Processing Order

The expanded schematic below shows how all events in the Cribl Stream ecosystem are processed linearly, from left to right.


Cribl Stream
ROUTES
ROUTE FILTERS
PROCESSING PIPELINES
SOURCES
Custom Command
Event Breakers
Fields
Pre-Processing Pipelines (Optional)
Post-Processing Pipelines (Optional)
DESTINATIONS
Collector 1
Source 2
Source 3
Source 4
PQ
Input Filter
Pack
Destination 1
Destination 2
Destination 3
Destination 4
Pre-process or normalize events from a Source
Post-process or normalize events to a Destination
Routes map events to Pipelines and Destinations
Processing Pipelines perform all event transformations
QUICKCONNECT
Bypass Routes, optionally inserting a Pipeline or Pack

Cribl Stream in great detail

Here are the stages of event processing:

1. Sources: Data arrives from your choice of external providers. (Cribl Stream 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 Stream'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. Persistent queue (Source-side): Optionally, you can enable PQ per Source instance, to disk-buffer and maximize data delivery despite downstream backpressure. Events will be written to a disk queue at this point.
8. Pre-processing Pipeline: Optionally, you can use a single Pipeline to condition (normalize) data from this input before the data proceeds further.

9. 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. (QuickConnect bypasses Routes.)
10. Processing Pipelines perform most 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. (QuickConnect optionally bypasses processing Pipelines.)
11. Post-processing Pipeline: Optionally, you can append a Pipeline to condition (normalize) data on its way into its Destination.
12. Persistent queue (Destination-side): Optionally, you can enable PQ per Destination instance, to disk-buffer and maximize data delivery during imbalances between inbound and outbound data rates. Events will be written to a disk queue at this point.
13. Destinations: Each Route/Pipeline combination forwards processed data to your choice of streaming or storage Destination. (Cribl Stream 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.

13.3. Routes

Before incoming events are transformed by a processing Pipeline, Cribl Stream 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 Stream’s QuickConnect rapid visual configuration tool as an alternative to Routes.

Accessing Routes

Select **Routing** > **Data Routes** from Cribl Stream’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

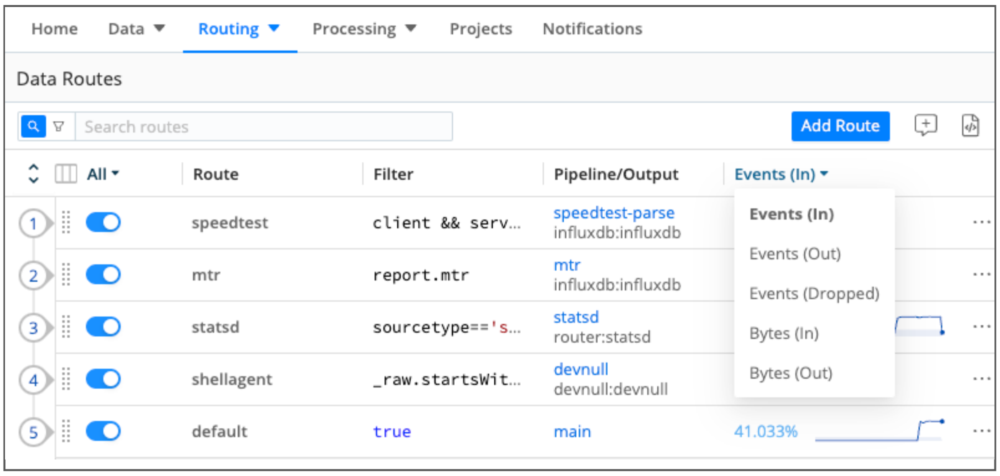
Routes apply filter expressions on incoming events to send matching results to the appropriate Pipeline. Filters are JavaScript-syntax–compatible expressions that are configured with each Route. Examples are:

- `true`
- `source=='foo.log' && fieldA=='bar'`

There can be multiple Routes in a Cribl Stream 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.



	All	Route	Filter	Pipeline/Output	Events (In)
1		speedtest	`client && serv…`	speedtest-parse influxdb:influxdb	
2		mtr	`report.mtr`	mtr influxdb:influxdb	
3		statsd	`sourcetype=='s…`	statsd router:statsd	
4		shellagent	`_raw.startsWit…`	devnull devnull:devnull	
5		default	`true`	main	41.033%

Routes and bytes

In the example above, incoming events will be evaluated first against the Route named **speedtest**, then against **mtr**, then against **statsd**, and so on. At the end, the **default** Route serves as a catch-all for any event that does not match any of the other Routes.

Above, note the **Events In** drop-down menu to toggle between displaying Events versus Bytes, and to display In, Out, or Dropped.

Displayed percentages of Events and Bytes In and Out are by comparison to throughput or filtering across all Routes. For example, if you have four Routes that share an identical filter, each Route's Events In might show approximately `25%` of the total inbound traffic across all Routes.

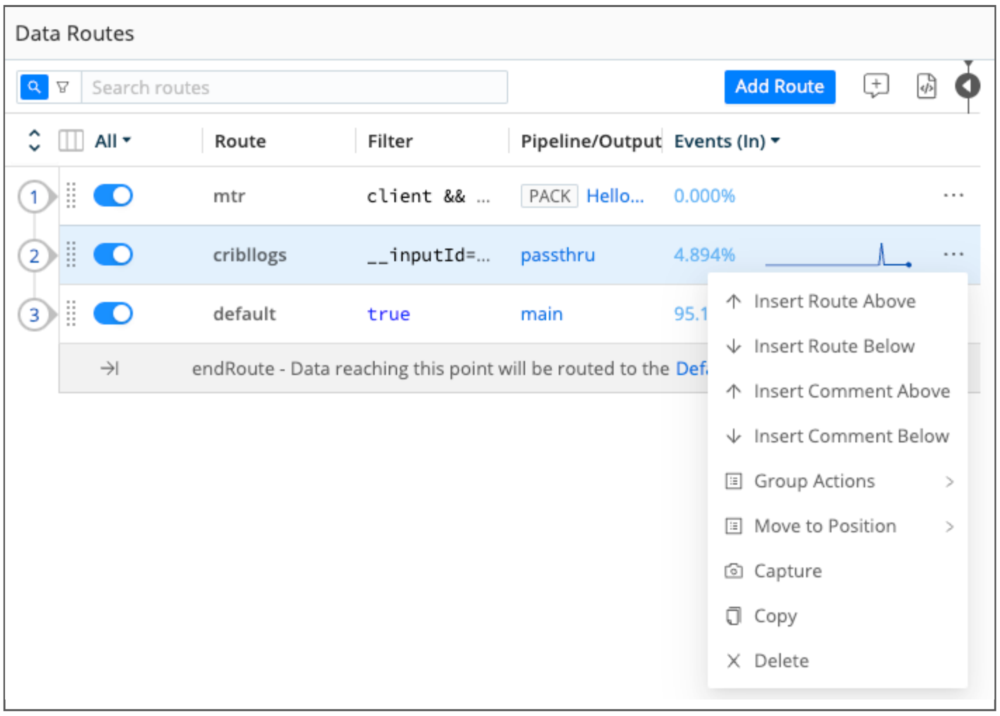
Managing the Routes Page

To apply a Route before another, simply drag it vertically. Use the toggles to turn Routes `On/Off` inline, as necessary, to facilitate development and debugging.

You can press the `]` (right-bracket) shortcut key to toggle between the Preview pane and the expanded Routes display shown above. (This works when no field has focus.)

Click a Route's Options (•••) menu to display multiple options:

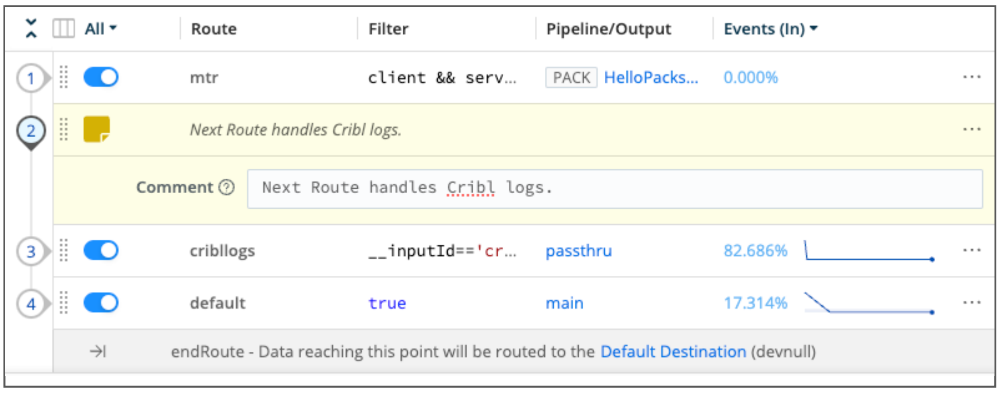
- Insert, group, move, copy, or delete Routes.
- Insert comments above or below Routes.
- Capture sample data through a selected Route.



Route > Options menu

Copying a Route displays a confirmation message and adds a Paste button next to **Add Route**. Pasting creates an exact duplicate of the Route, with a warning indicator to change its duplicate name.

Comments work like a Pipeline's Comment Functions: You can use them to describe Routes' purposes and/or interactions.

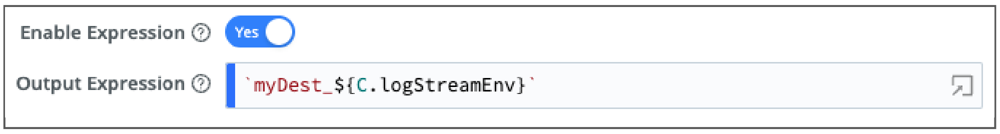

All
Route
Filter
Pipeline/Output
Events (In)
1 mtr client && serv... PACK HelloPacks... 0.000%
2 Next Route handles Cribl logs.
Comment Next Route handles Cribl logs.
3 cribllogs __inputId=='cr... passthru 82.686%
4 default true main 17.314%
endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Comments make the Routing table self-documenting

Output Destination

You can configure each Route with an **Output** that defines a Destination to send events to, after they're processed by the Pipeline.

If you toggle **Enable expression** to `Yes`, the **Output** field changes to an **Output expression** field. Here, you can enter a JavaScript expression that Cribl Stream will evaluate as the name of the Destination. Sample expression format: `` `myDest_${C.logStreamEnv}` ``. (This evaluation happens at Route construction time, not per event.)



Enable Expression: Yes

Output Expression: `` `myDest_${C.logStreamEnv}` ``

Output expression enabled

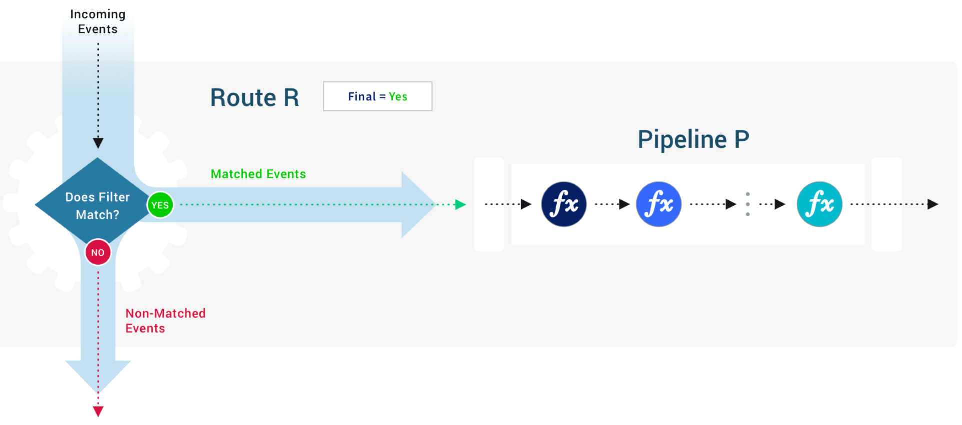
Expressions that match no Destination name will silently fail.

On Routes within Packs, neither **Output** control is available, because Packs cannot specify Destinations.

The `Final` Toggle

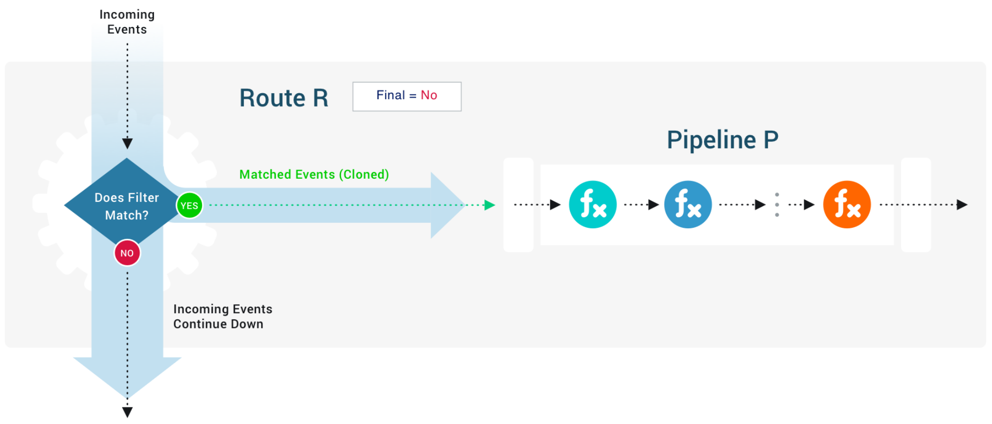
The `Final` toggle in Route settings controls what happens to an event after a Route-Pipeline pair matches it.

When `Final` is toggled to `Yes` (default), the Route "consumes" an event, meaning the event will not pass down to any other Route below.



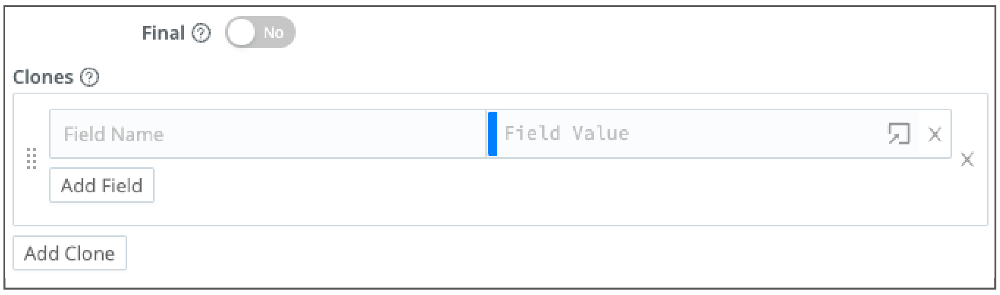
The Route is Final and non-matched events stop here

When `Final` is toggled to `No`, the event will be processed by this Route, **and** then passed on to the next Route below.



The Route isn't Final and all events pass further on

When you toggle `Final` to `No`, the event sent to the current Route technically becomes a *clone* of the original event. The **Add Clone** button lets you add a fields – names and values – to the cloned events sent to this Route's Pipeline.



Non-final Route: add clone fields

Event Cloning Strategy

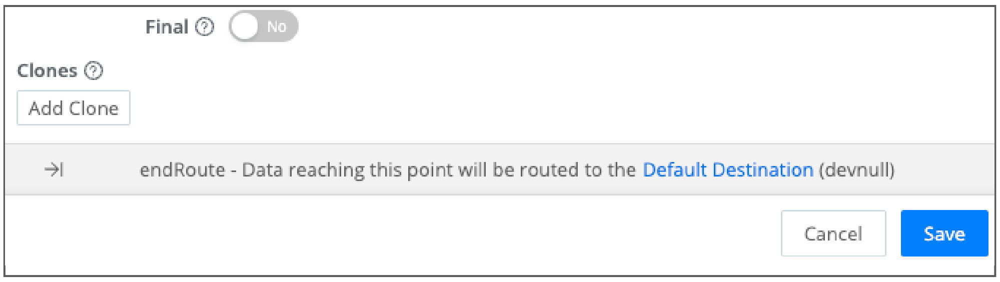
Depending on your cloning needs, you might want to follow a **most-specific first** or a **most-general first** processing strategy. The general goal is to minimize the number of filters/Routes an event gets evaluated against. For example:

If you don't need to clone any events (that is, `Final` is toggled to `Yes` everywhere), then it makes sense to start with the broadest expression at the top. This will consume as many events as early as possible.

If you need to clone a narrow set of events, then it might make sense to do that upfront. Then, follow it with a Route that consumes those clones immediately after.

The `endRoute` Bumper

The `endRoute` bumper appears at the bottom of the Routing table. This is a reminder that, if **no** Route in the table has the `Final` flag enabled, events will continue to Cribl Stream's configured Default Destination.


Final No

Clones

Add Clone

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

Cancel Save

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.

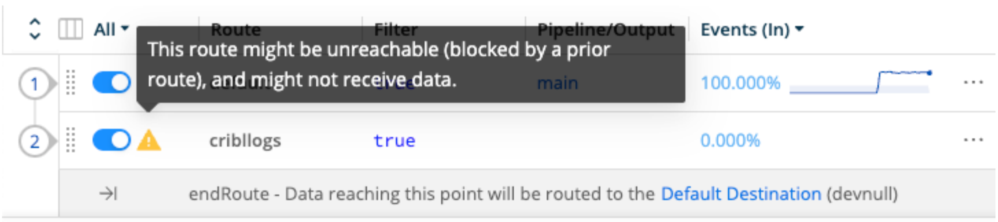
Route Groups

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.


All Route Filter Pipeline/Output Events (In)

This route might be unreachable (blocked by a prior route), and might not receive data.

1 main 100.000%

2 cribllogs true 0.000%

endRoute - Data reaching this point will be routed to the Default Destination (devnull)

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 (toggle is set to `On`).
- Previous Route is final (**Final** toggle is set to `Yes`).

- Previous Route’s **Filter** expression evaluates to true, (e.g., `true`, `1 === 1`, etc.).

Note that the third condition above can be triggered intermittently by a randomizing method like `Math.random()`. This might be included in a previous Route’s own Filter expression, or in a Pipeline Function (such as one configured for random data sampling).

Routing with Output Router

Output Router Destinations offer another way to route data. These function as meta-Destinations, in that they allow you to send data to multiple peer Destinations based on rules. Rules are evaluated in order, top->down, with the first match being the winner.

13.4. Pipelines

Data matched by a given Route is delivered to a Pipeline. Pipelines are the heart of Cribl Stream 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

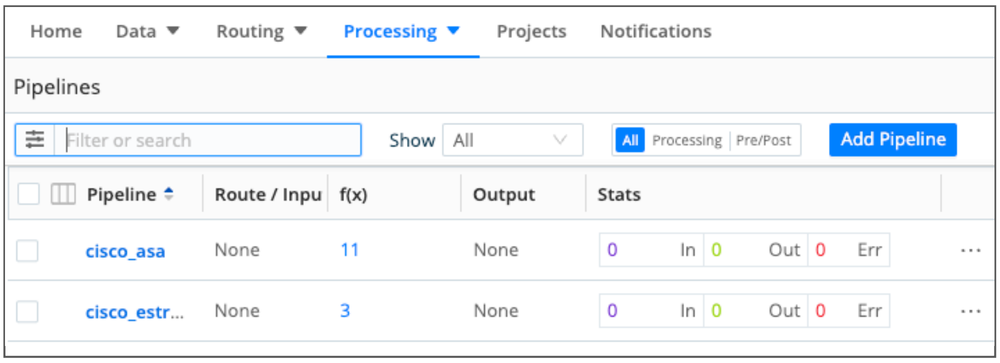
Select **Processing** > **Pipelines** from Cribl Stream's global top nav (single-instance deployments) or from a Worker Group's/Fleet's top nav (distributed deployments). Next, click any displayed Pipeline to see or reconfigure its contained Functions.

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 Stream 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 Stream'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.)



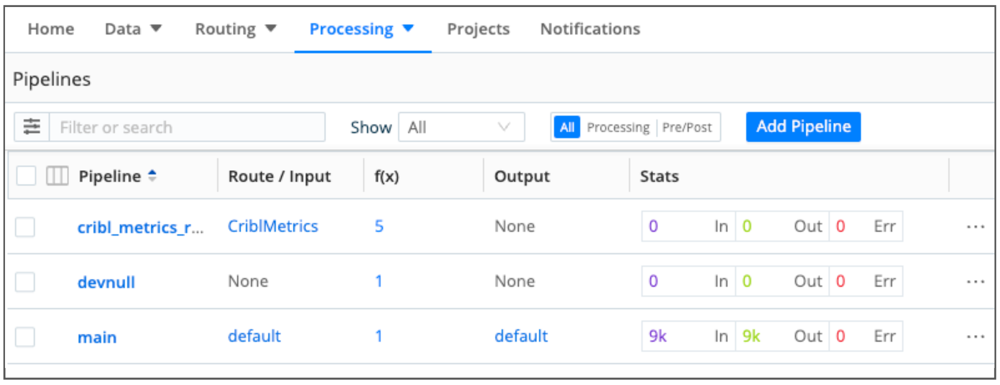
Pipeline	Route / Inpu	f(x)	Output	Stats
cisco_asa	None	11	None	0 In 0 Out 0 Err
cisco_estr...	None	3	None	0 In 0 Out 0 Err

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.

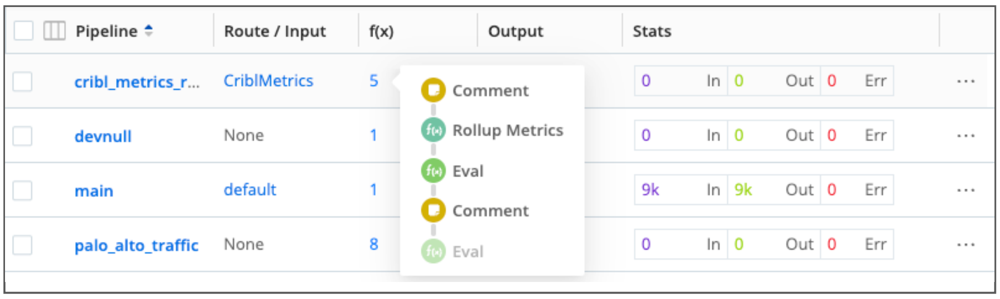


Pipeline	Route / Input	f(x)	Output	Stats
cribl_metrics_r...	CriblMetrics	5	None	0 In 0 Out 0 Err
devnull	None	1	None	0 In 0 Out 0 Err
main	default	1	default	9k In 9k Out 0 Err

Pipelines and Route inputs

You can press the] (right-bracket) shortcut key to toggle between the Preview pane and an expanded Pipelines display. (This shortcut works when no field has focus.)

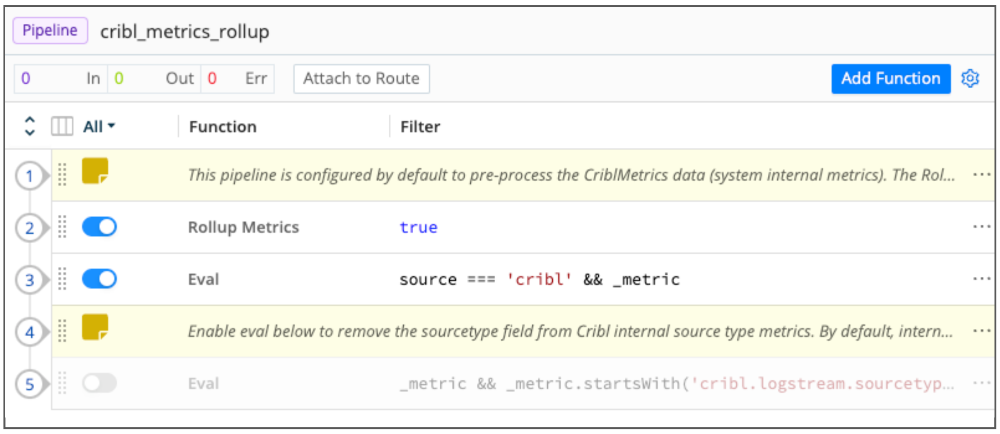
In the condensed Pipelines display above, you can also hover over any Pipeline's **Functions** column to see a vertical preview of the stack of Functions contained in the Pipeline:



Pipeline	Route / Input	f(x)	Output	Stats
cribl_metrics_r...	CriblMetrics	5		0 In 0 Out 0 Err
devnull	None	1		0 In 0 Out 0 Err
main	default	1		9k In 9k Out 0 Err
palo_alto_traffic	None	8		0 In 0 Out 0 Err

Preview on hovering over the top Pipeline (highlighted in gray)

Within the Pipeline, events are processed by each Function, in order. A Pipeline will always move events in the direction that points outside of the system. This is on purpose, to keep the design simple and avoid potential loops.


Pipeline cribl_metrics_rollup

0 In 0 Out 0 Err | Attach to Route | Add Function

	Function	Filter
1	*This pipeline is configured by default to pre-process the CriblMetrics data (system internal metrics). The Rol...*	
2	Rollup Metrics	`true`
3	Eval	`source === 'cribl' && _metric`
4	*Enable eval below to remove the sourcetype field from Cribl internal source type metrics. By default, intern...*	
5	Eval	`_metric && _metric.startsWith('cribl.logstream.sourcetyp...`

Pipeline Functions

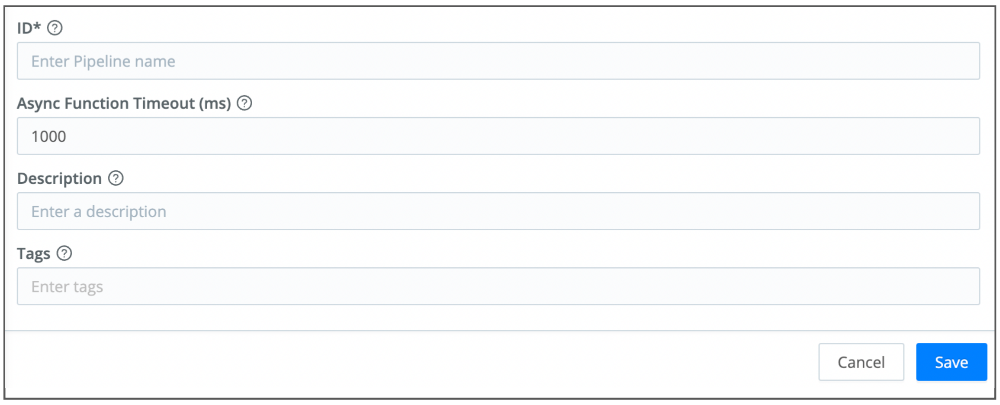
Use the **Attach to Route** link at upper left to associate a new Pipeline with a Route.

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.


ID*: Enter Pipeline name

Async Function Timeout (ms): 1000

Description: Enter a description

Tags: Enter tags

Cancel | Save

Pipeline Settings

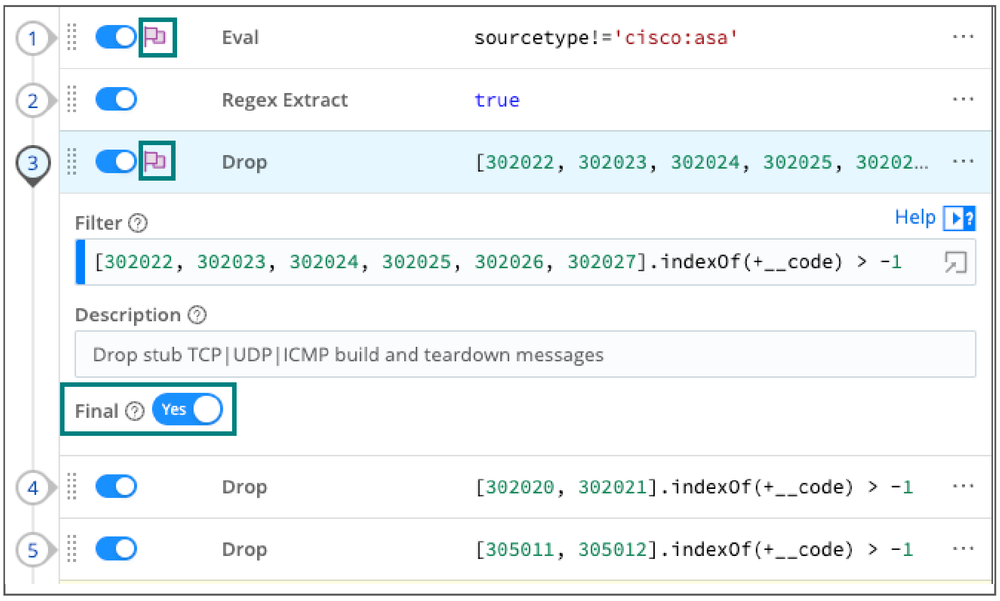
The `Final` Toggle

The `Final` toggle in Function settings controls what happens to the results of a Function.

When `Final` is toggled to `No` (default), events will be processed by this Function, **and** then passed on to the next Function below.

When `Final` is toggled to `Yes`, the Function "consumes" the results, meaning they will not pass down to any other Function below.

A flag in Pipeline view indicates that a Function is `Final`.

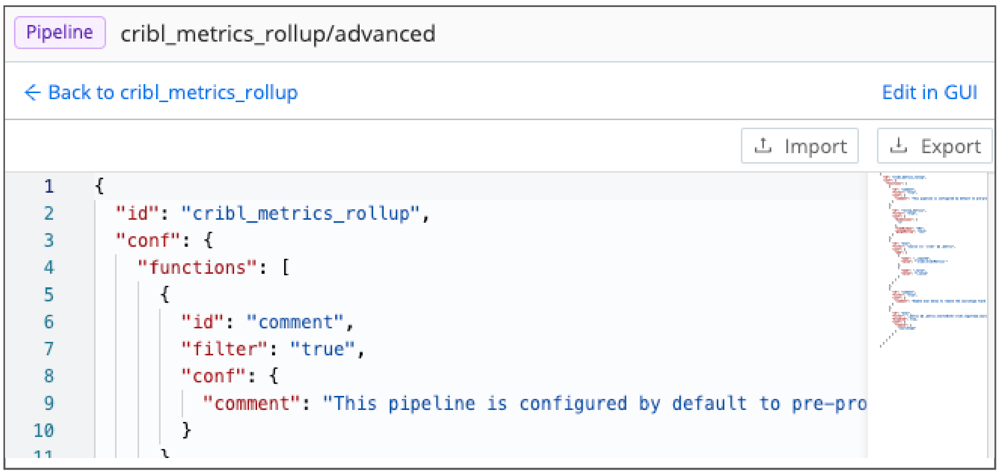


```
1  Eval            sourcetype!='cisco:asa'
2  Regex Extract   true
3  Drop            [302022, 302023, 302024, 302025, 30202…
   Filter
   [302022, 302023, 302024, 302025, 302026, 302027].indexOf(+__code) > -1
   Description
   Drop stub TCP|UDP|ICMP build and teardown messages
   Final  Yes
4  Drop            [302020, 302021].indexOf(+__code) > -1
5  Drop            [305011, 305012].indexOf(+__code) > -1
```

The Eval and first Drop Functions bearing the Final flag

Advanced Mode (JSON Editor)

Once you've clicked the gear button to enter Pipeline Settings, you can click **Manage as JSON** at the upper right to edit the Pipeline's definition in a JSON text editor. In this mode's editor, you can directly edit multiple values. You can also use the **Import** and **Export** buttons here to copy and modify existing Pipeline configurations, as `.json` files.



```
{
  "id": "cribl_metrics_rollup",
  "conf": {
    "functions": [
      {
        "id": "comment",
        "filter": "true",
        "conf": {
          "comment": "This pipeline is configured by default to pre-pro
        }
```

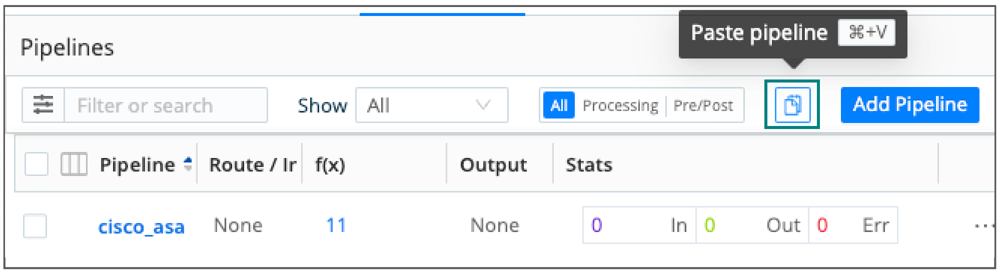
Advanced Pipeline Editing

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.

Copying a Pipeline displays a confirmation message and the (highlighted) Paste button shown below.



Paste button for copied Pipeline

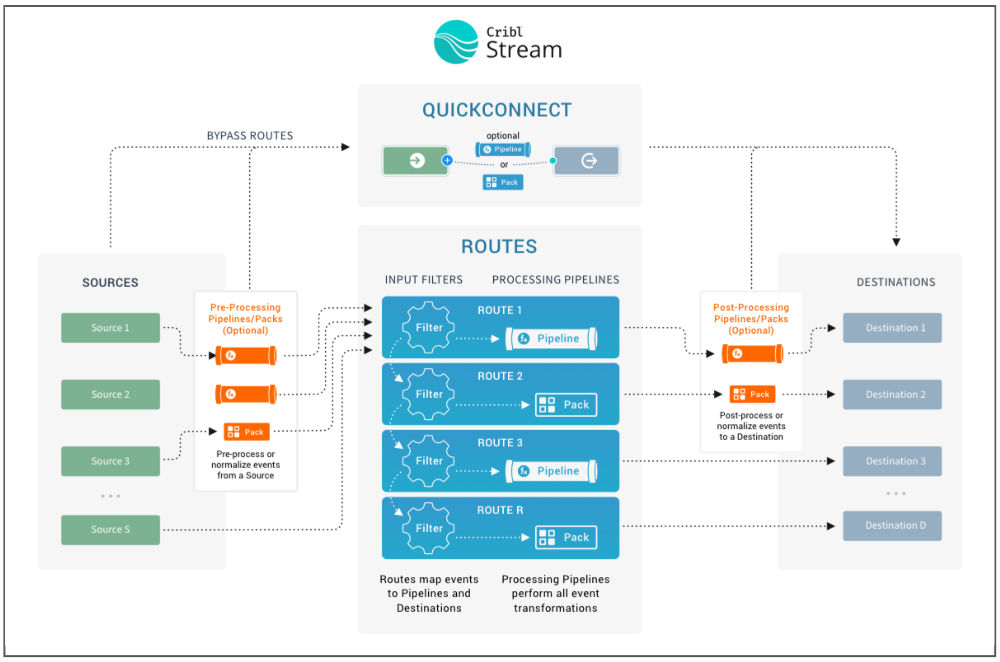
Pasting prompts you to confirm, or change, a modified name for the new Pipeline. The result will be an exact duplicate of the original Pipeline in all but name.

Chaining Pipelines

You can use the Chain Function to send the output of a Pipeline to another Pipeline or Pack. There are scope restrictions within Packs, and general guardrails against circular references.

Types of Pipelines

You can apply various Pipeline types at different stages of data flow. All Pipelines have the same basic internal structure (a series of Functions) – the types below differ only in their position in the system.


Cribl Stream
QUICKCONNECT
optional
Pipeline
or
Pack
BYPASS ROUTES
ROUTES
INPUT FILTERS
PROCESSING PIPELINES
SOURCES
Source 1
Source 2
Source 3
Source S
Pre-Processing Pipelines/Packs (Optional)
Pre-process or normalize events from a Source
ROUTE 1
ROUTE 2
ROUTE 3
ROUTE R
Filter
Routes map events to Pipelines and Destinations
Processing Pipelines perform all event transformations
Post-Processing Pipelines/Packs (Optional)
Post-process or normalize events to a Destination
DESTINATIONS
Destination 1
Destination 2
Destination 3
Destination D

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 Stream 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.

13.5. Packs

Packs enable Cribl Stream administrators and developers to pack up and share complex configurations and workflows across multiple Worker Groups, or across organizations.

Packs = Portability

With a Cribl Stream 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 Stream environment.

For example, where a Pipeline's configuration references Lookup file(s), Cribl Stream 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 Stream instances. If you import the Lookups into the Pack, you can develop and test a configuration, and then port it from development to production instances, or readily deploy it to multiple Worker Groups.

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 Stream ecosystem.

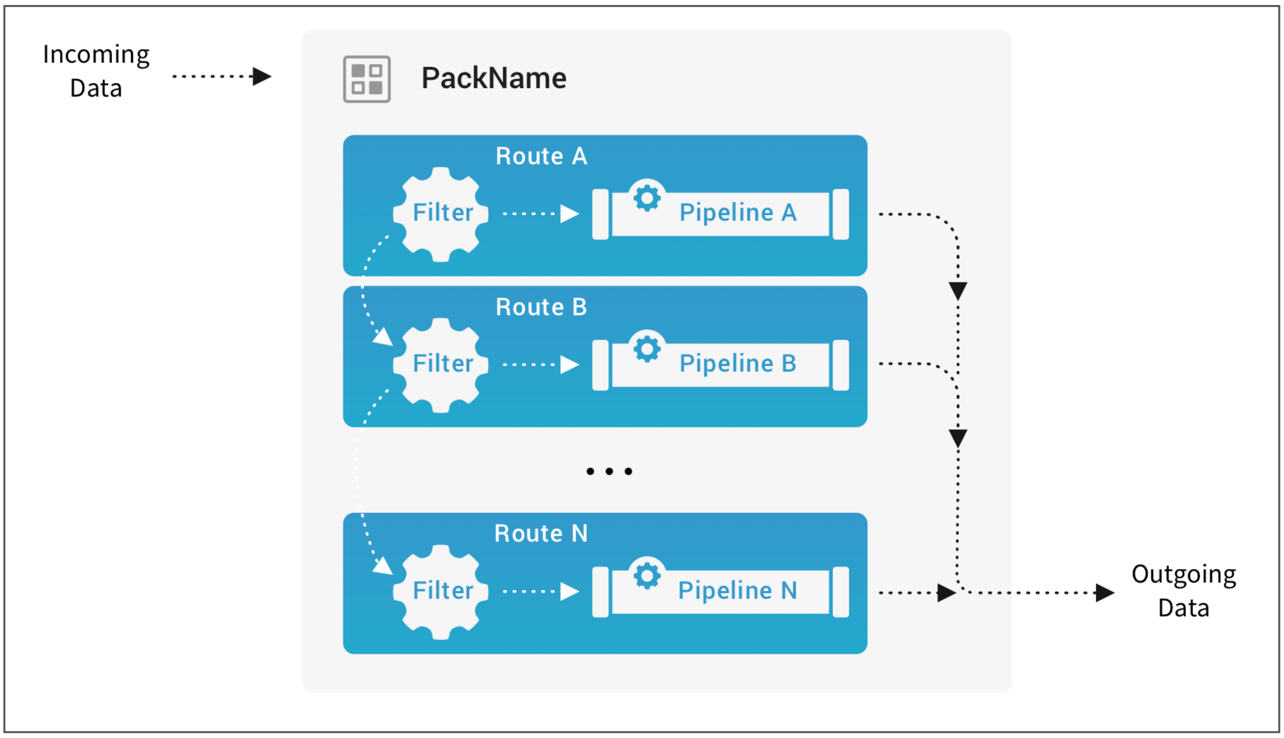
What Is a Pack?

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 Knowledge objects (Lookups, Global Variables, etc.) that are defined outside the Pack. Knowledge objects defined **within** a Pack are not available to Cribl Stream resources outside the Pack. In either scenario, you must explicitly duplicate the Knowledge objects in the opposite scope.

What's Not in a Pack?

Sources, Collectors, and Destinations are external to Packs, so you can't specify them within a Pack. This excludes a few other things:

- Routes configured within a Pack can't specify a Destination.
- Packs can't include Event Breakers, which are associated with Sources. (However, you can instead use the Event Breaker Function in 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 Stream 4.0.x cannot be used in any pre-4.0 version of Cribl Stream. (If you try, you'll see a `should NOT have additional properties` error.) To avoid this problem, Cribl recommends that you upgrade to Cribl Stream 4.0.x or later.

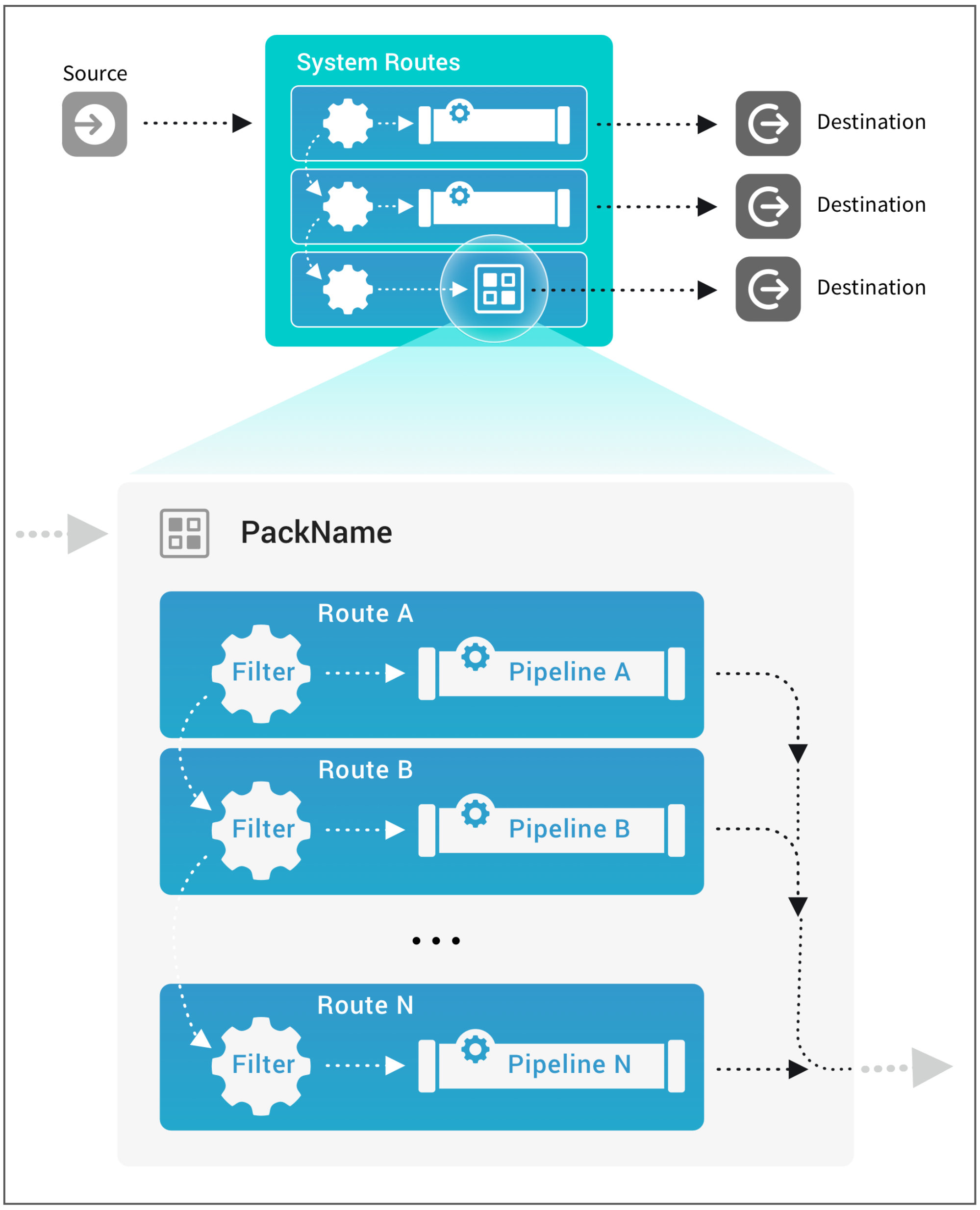
For compatibility questions about any individual Packs, please contact us in Cribl Community Slack's `#packs` channel.

Where Can I Use Packs?

Wherever you can reference a Pipeline, you can specify a Pack:

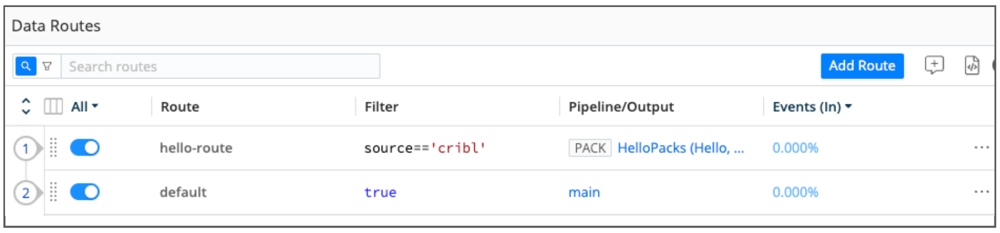
- In Sources, where you attach pre-processing Pipelines.
- In Destinations, where you attach post-processing Pipelines.
- In Routes, in the Routing table's **Pipeline/Output** column.

This expanded view shows how a Pack can replace a Pipeline in a Route:



A Pack snaps into Cribl Stream like an enhanced Pipeline

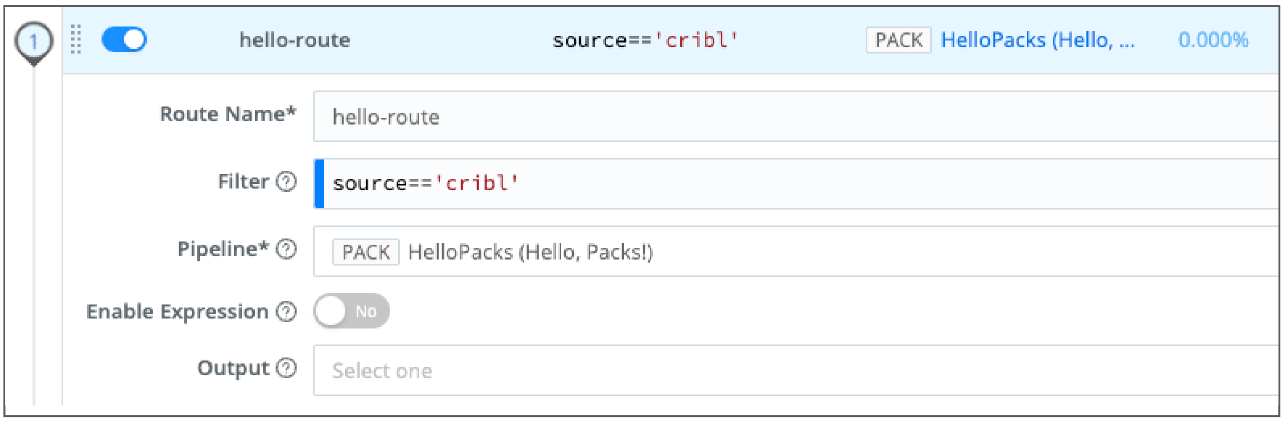
Packs are distinguished in the display with a **PACK** badge, as you can see here in the Routing table:



	All	Route	Filter	Pipeline/Output	Events (In)
1		hello-route	source=='cribl'	PACK HelloPacks (Hello, ...	0.000%
2		default	true	main	0.000%

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:


hello-route source=='cribl' PACK HelloPacks (Hello, ... 0.000%

Route Name*: hello-route

Filter: source=='cribl'

Pipeline*: PACK HelloPacks (Hello, Packs!)

Enable Expression: No

Output: Select one

PACK badge on a Pack connected to a Route

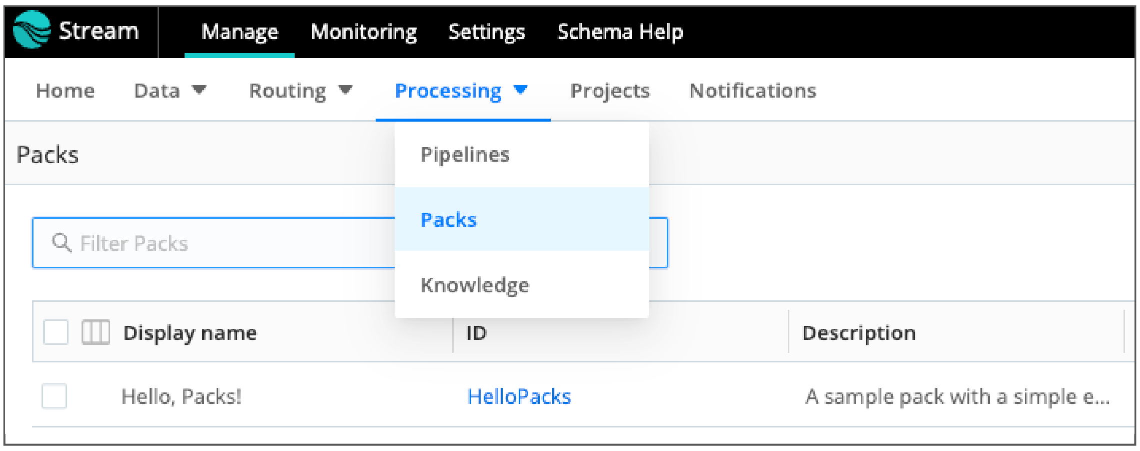
Cribl Stream’s **Monitoring** page includes a **Packs** link where you can monitor Packs’ throughput.

Accessing Packs

You access Packs differently, depending on your deployment type.

Single-Instance Deployment

In a single-instance deployment, Packs are global. From Cribl Stream’s top-level navigation, just select **Processing** > **Packs**. On the filesystem, Packs (including those that you add) are stored at `$CRIBL_HOME/default/`.

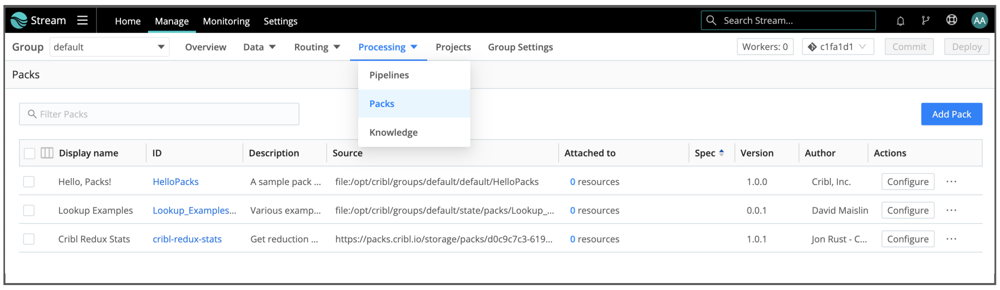


Display name	ID	Description
Hello, Packs!	HelloPacks	A sample pack with a simple e...

Packs, single-instance navigation

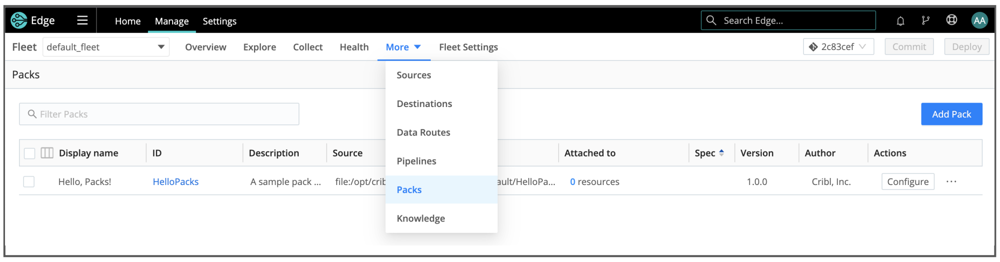
Distributed Deployment

In a distributed deployment, Packs are associated with (and installed within) Worker Groups. Select **Manage**, and then click into the Worker Group you want to manage (for example, `default`). Next, from that Worker Group’s top nav, select **Processing** > **Packs** (Stream) or **More** > **Packs** (Edge).



Display name	ID	Description	Source	Attached to	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack ...	file:/opt/cribl/groups/default/default/HelloPacks	0 resources		1.0.0	Cribl, Inc.	Configure ···
Lookup Examples	Lookup_Examples...	Various examp...	file:/opt/cribl/groups/default/state/packs/Lookup_...	0 resources		0.0.1	David Maislin	Configure ···
Cribl Redux Stats	cribl-redux-stats	Get reduction ...	https://packs.cribl.io/storage/packs/d0c9c7c3-619...	0 resources		1.0.1	Jon Rust - C...	Configure ···

Stream Group > Manage Packs page



Display name	ID	Description	Source	Attached to	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack ...	file:/opt/crib ault/HelloPa...	0 resources		1.0.0	Cribl, Inc.	Configure ···

Edge Fleet > Manage Packs page

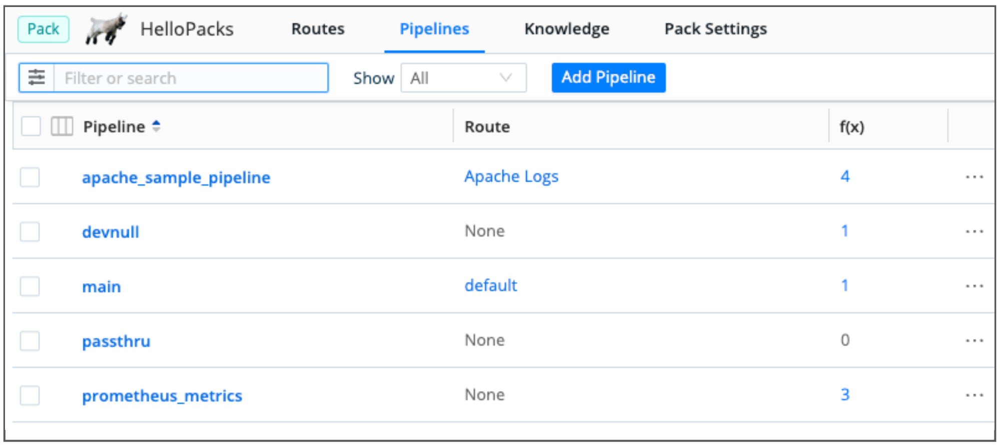
Each Group's Packs are stored at at `$CRIBL_HOME/groups/<group-name>/default/`. (In a typical installation, you'll find the starter `HelloPacks` Pack at `/opt/cribl/groups/default/default/`.)

By design, you can readily share Packs **across** Worker Groups by copying them between each other.

Getting Started with Packs

To unpack Packs, use the above instructions (per deployment type) to navigate to the **HelloPacks** example Pack shipped with Cribl Stream. On the **Manage Packs** page, click this Pack's row to see its configuration.

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 Stream's global level. This Pack also includes an Apache-specific sample Pipeline – click it to unpack that, too.



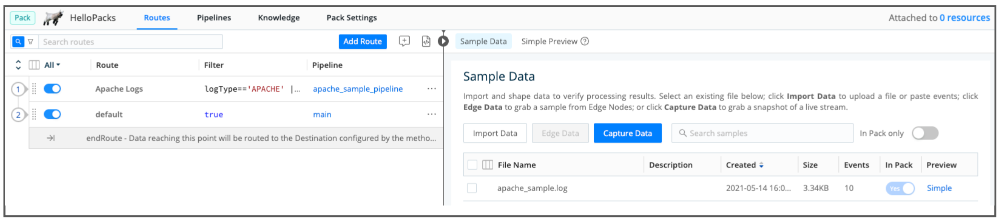
Pipeline	Route	f(x)
apache_sample_pipeline	Apache Logs	4
devnull	None	1
main	default	1
passthru	None	0
prometheus_metrics	None	3

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 Stream's top nav.



	Route	Filter	Pipeline
1	Apache Logs	`logType=='APACHE' |...`	apache_sample_pipeline
2	default	`true`	main

endRoute - Data reaching this point will be routed to the Destination configured by the metho...

Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a snapshot of a live stream.

File Name	Description	Created	Size	Events	In Pack	Preview
apache_sample.log		2021-05-14 16:0...	3.34KB	10	Yes	Simple

Configuring a Pack

The left pane's submenu links give you access to configuration objects specific to this Pack.

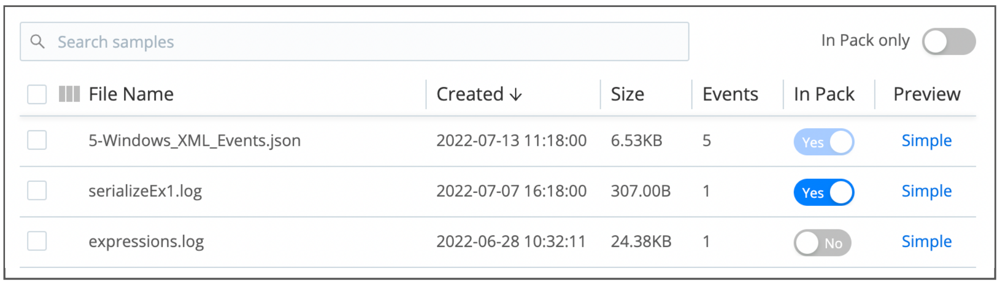
Sample Data

The right pane defaults to displaying all sample data files available on your Cribl Stream instance. If you prefer to filter only sample files internal to the Pack, toggle **In Pack only** 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 Stream, 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 Stream. It is available to this Pack. Toggle this to `Yes` (dark blue) if you want the sample file to export along with the Pack.



File Name	Created	Size	Events	In Pack	Preview
5-Windows_XML_Events.json	2022-07-13 11:18:00	6.53KB	5	Yes	Simple
serializeEx1.log	2022-07-07 16:18:00	307.00B	1	Yes	Simple
expressions.log	2022-06-28 10:32:11	24.38KB	1	No	Simple

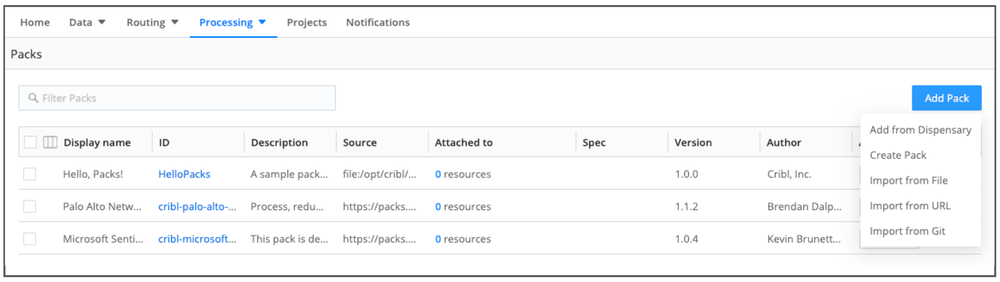
Sample file in Pack

Basically, you can manipulate all the options here as you'd work with their big sister or brother in Cribl Stream'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.



Display name	ID	Description	Source	Attached to	Spec	Version	Author
Hello, Packs!	HelloPacks	A sample pack...	file:/opt/cribl/...	0 resources		1.0.0	Cribl, Inc.
Palo Alto Netw...	cribl-palo-alto-...	Process, redu...	https://packs....	0 resources		1.1.2	Brendan Dalp...
Microsoft Senti...	cribl-microsoft...	This pack is de...	https://packs....	0 resources		1.0.4	Kevin Brunett...

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.

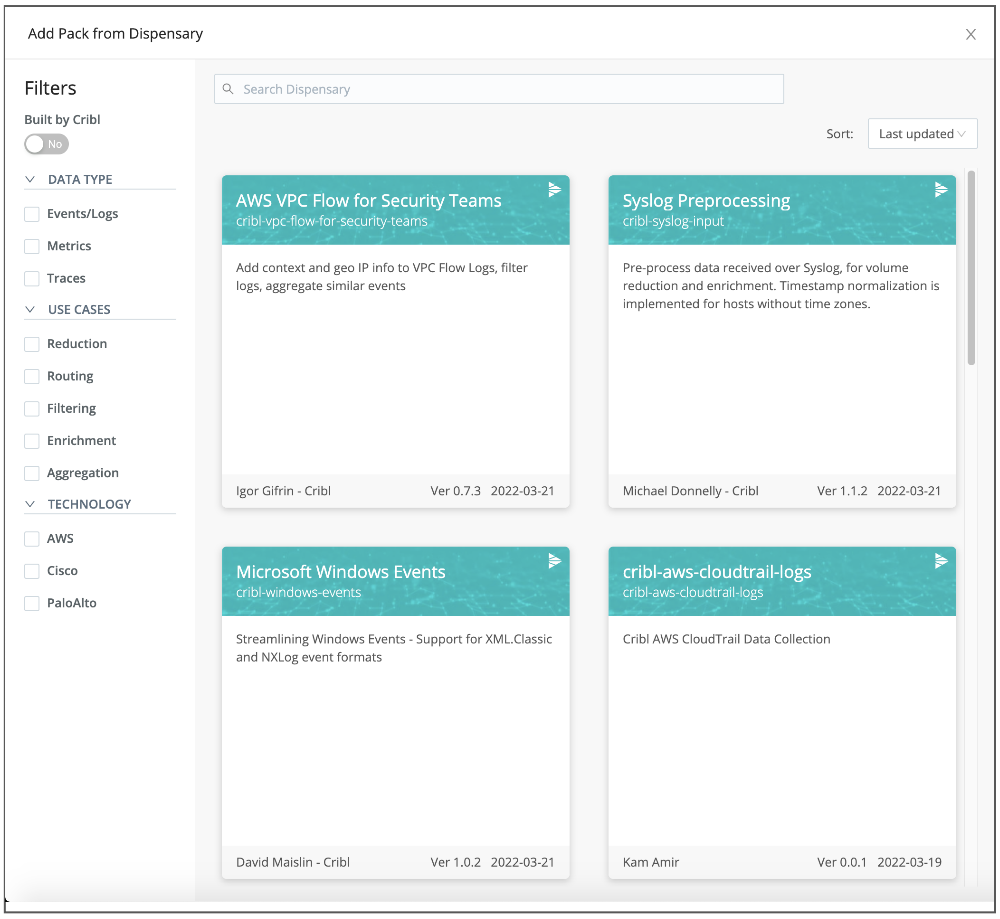
As an additional protection layer, all Pack import modals provide an **Allow custom functions** toggle. In the toggle's default `No` position, if Cribl Stream detects custom functions in the specified Pack, it will block the import with an error message. If you trust the Pack's provider,set the toggle 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 Stream, 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 Stream 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 Stream's UI, as outlined in Add from Packs Dispensary below.


Add Pack from Dispensary

Filters

Built by Cribl: No

DATA TYPE
- Events/Logs
- Metrics
- Traces

USE CASES
- Reduction
- Routing
- Filtering
- Enrichment
- Aggregation

TECHNOLOGY
- AWS
- Cisco
- PaloAlto

Search Dispensary

Sort: Last updated

AWS VPC Flow for Security Teams
cribl-vpc-flow-for-security-teams
Add context and geo IP info to VPC Flow Logs, filter logs, aggregate similar events
Igor Gifrin - Cribl Ver 0.7.3 2022-03-21

Syslog Preprocessing
cribl-syslog-input
Pre-process data received over Syslog, for volume reduction and enrichment. Timestamp normalization is implemented for hosts without time zones.
Michael Donnelly - Cribl Ver 1.1.2 2022-03-21

Microsoft Windows Events
cribl-windows-events
Streamlining Windows Events - Support for XML.Classic and NXLog event formats
David Maislin - Cribl Ver 1.0.2 2022-03-21

cribl-aws-cloudtrail-logs
cribl-aws-cloudtrail-logs
Cribl AWS CloudTrail Data Collection
Kam Amir Ver 0.0.1 2022-03-19

Cribl Packs Dispensary™ (as displayed in Cribl Stream's **Add** drawer)

Interested in publishing your own Packs on the Cribl Packs Dispensary™? See Publishing a Pack.

Add from Packs Dispensary

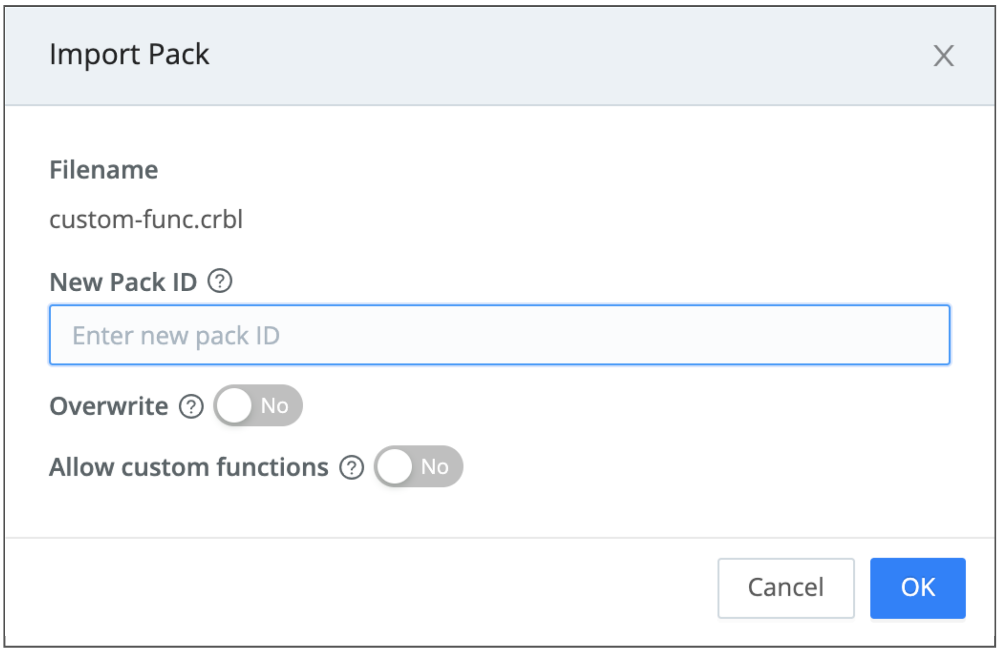
To add a Pack from the Cribl Packs Dispensary™ sharing site:

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.

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.


Import Pack

Filename
custom-func.crbl

New Pack ID
Enter new pack ID

Overwrite: No

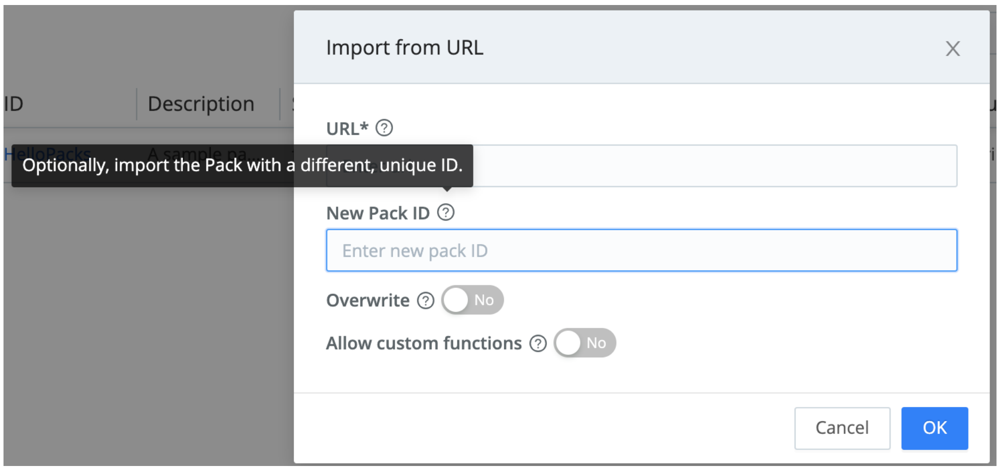
Allow custom functions: No

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.



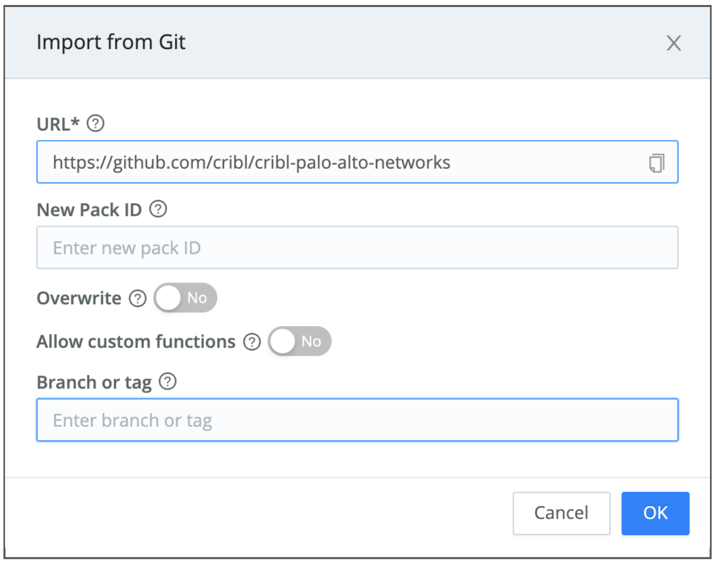
Confirming file import from URL

To import a Pack from a public URL, Cribl Stream'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.


Import from Git

URL* https://github.com/cribl/cribl-palo-alto-networks

New Pack ID: Enter new pack ID

Overwrite: No

Allow custom functions: No

Branch or tag: Enter branch or tag

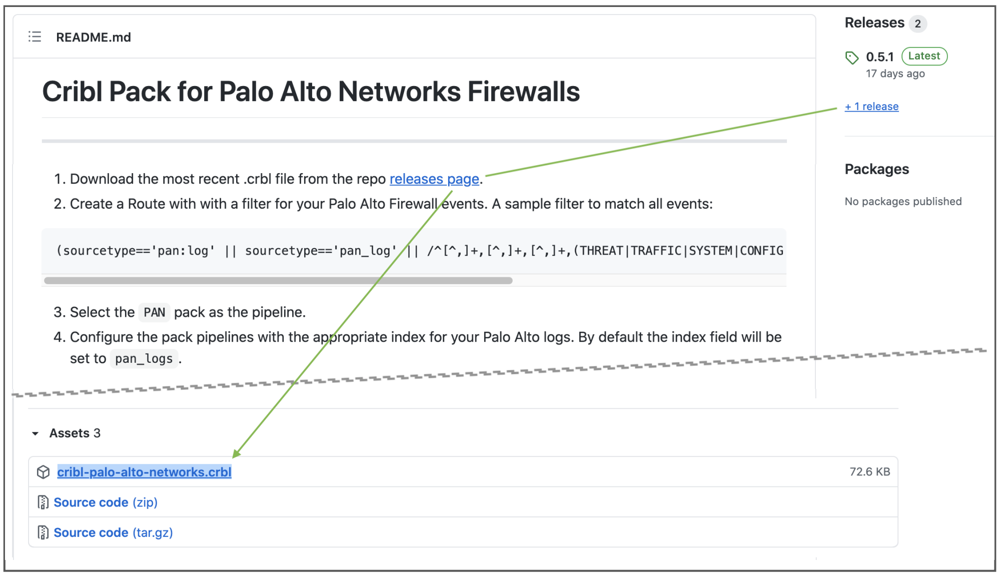
Importing from a Git repo

To import a Pack from a public repo, Cribl Stream'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 Stream'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 Stream.


README.md

Cribl Pack for Palo Alto Networks Firewalls

1. Download the most recent .crbl file from the repo releases page.
2. Create a Route with with a filter for your Palo Alto Firewall events. A sample filter to match all events:

```
(sourcetype=='pan:log' || sourcetype=='pan_log' || /^[^,]+,[^,]+,[^,]+,(THREAT|TRAFFIC|SYSTEM|CONFIG
```

3. Select the `PAN` pack as the pipeline.
4. Configure the pack pipelines with the appropriate index for your Palo Alto logs. By default the index field will be set to `pan_logs`.

Releases 2

0.5.1 Latest
17 days ago

+ 1 release

Packages

No packages published

Assets 3

cribl-palo-alto-networks.crbl	72.6 KB
Source code (zip)	
Source code (tar.gz)	

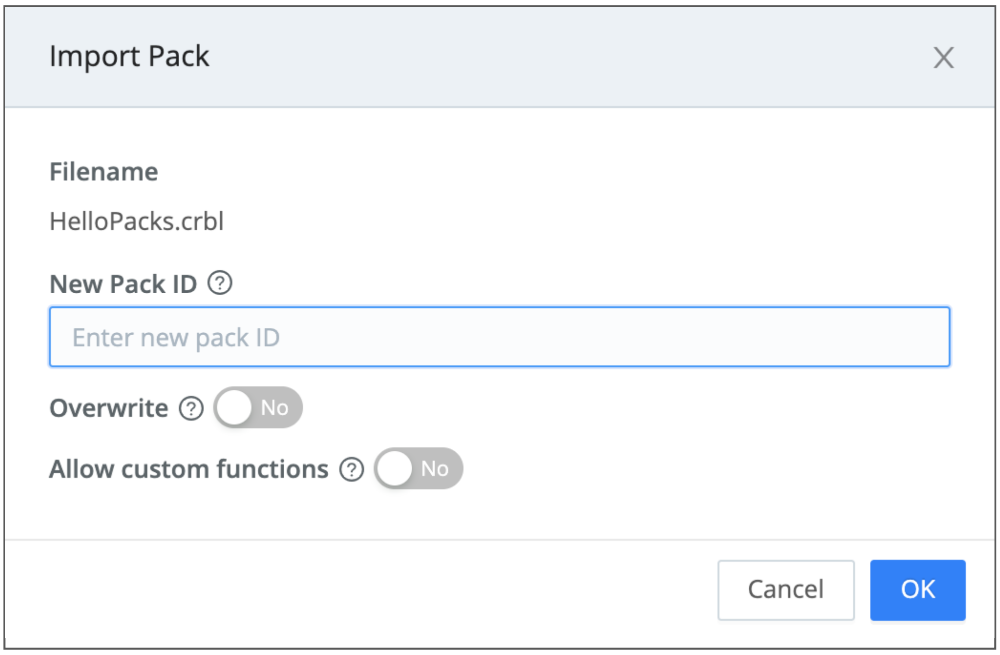
Downloading a `.crbl` file from the Cribl Pack Dispensary's Web UI

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 Worker Group (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.


Import Pack

Filename

HelloPacks.crbl

New Pack ID

Enter new pack ID

Overwrite: No

Allow custom functions: No

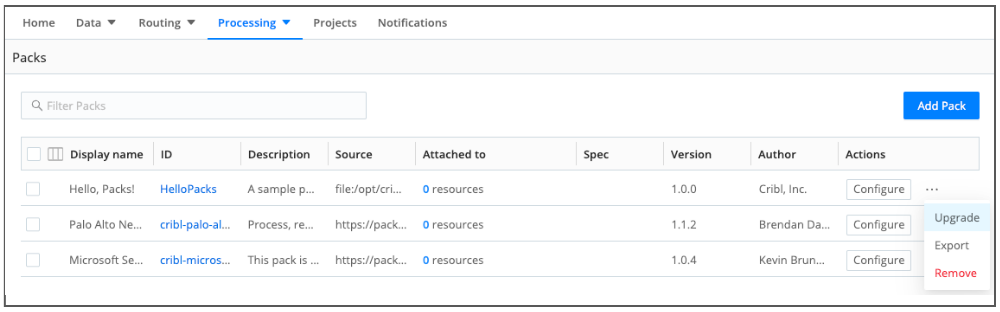
Renaming a Pack on import

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 Stream 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 menu icon on its row and select **Upgrade**.



Display name	ID	Description	Source	Attached to	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample p...	file:/opt/cri...	0 resources		1.0.0	Cribl, Inc.	Configure
Palo Alto Ne...	cribl-palo-al...	Process, re...	https://pack...	0 resources		1.1.2	Brendan Da...	Configure
Microsoft Se...	cribl-micros...	This pack is ...	https://pack...	0 resources		1.0.4	Kevin Brun...	Configure

Upgrading an existing Pack

If you've modified an installed Pack, Cribl Stream 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.

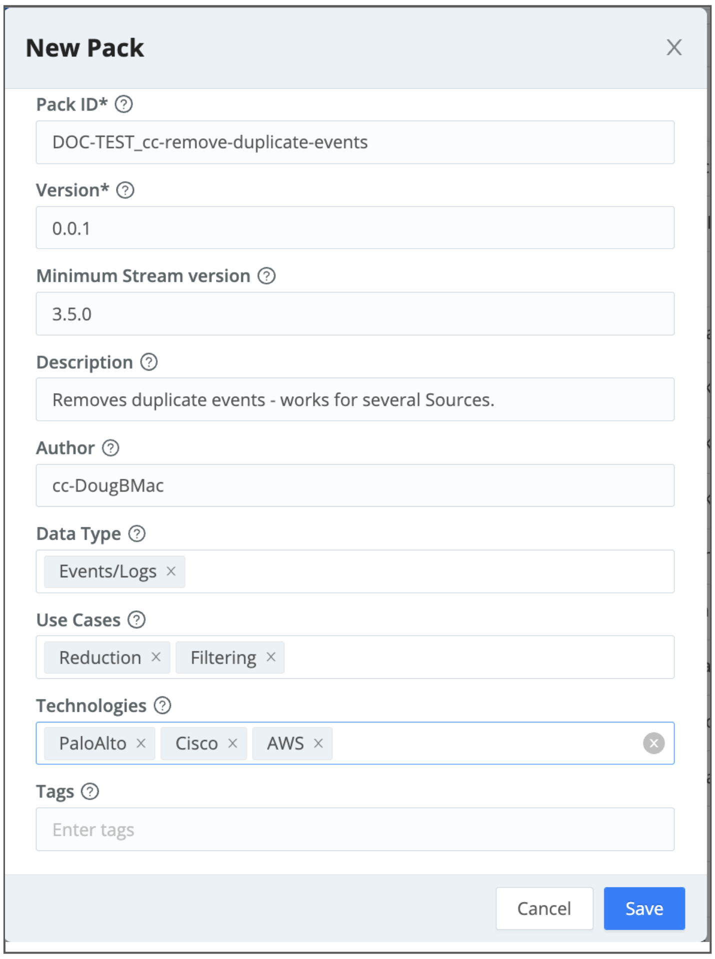
Creating a Pack

You can create a new Pack from scratch, to consolidate and export multiple Cribl Stream 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 Stream instance must have a separate **Pack ID**, but you can assign arbitrary **Display names**.
- **Version** is a required field identifying the Pack's own versioning.
- **Minimum Stream version** is an optional field specifying the lowest compatible version of Cribl Stream 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**.


New Pack

Pack ID*: DOC-TEST_cc-remove-duplicate-events

Version*: 0.0.1

Minimum Stream version: 3.5.0

Description: Removes duplicate events - works for several Sources.

Author: cc-DougBMac

Data Type: Events/Logs

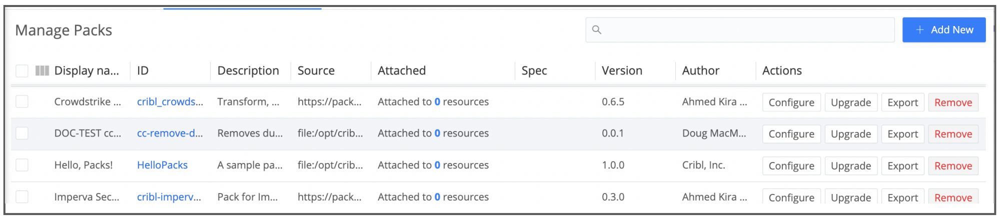
Use Cases: Reduction, Filtering

Technologies: PaloAlto, Cisco, AWS

Tags: Enter tags

Creating a Pack

6. On the **Manage Packs** page, click the new Pack's row to open the Pack.


Manage Packs

Display na...	ID	Description	Source	Attached	Spec	Version	Author	Actions
Crowdstrike ...	cribl_crowds...	Transform, ...	https://pack...	Attached to 0 resources		0.6.5	Ahmed Kira ...	Configure Upgrade Export Remove
DOC-TEST cc...	cc-remove-d...	Removes du...	file:/opt/crib...	Attached to 0 resources		0.0.1	Doug MacM...	Configure Upgrade Export Remove
Hello, Packs!	HelloPacks	A sample pa...	file:/opt/crib...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
Imperva Sec...	cribl-imperv...	Pack for Im...	https://pack...	Attached to 0 resources		0.3.0	Ahmed Kira ...	Configure Upgrade Export Remove

Manage Packs page

7. Use the standard Cribl Stream 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.

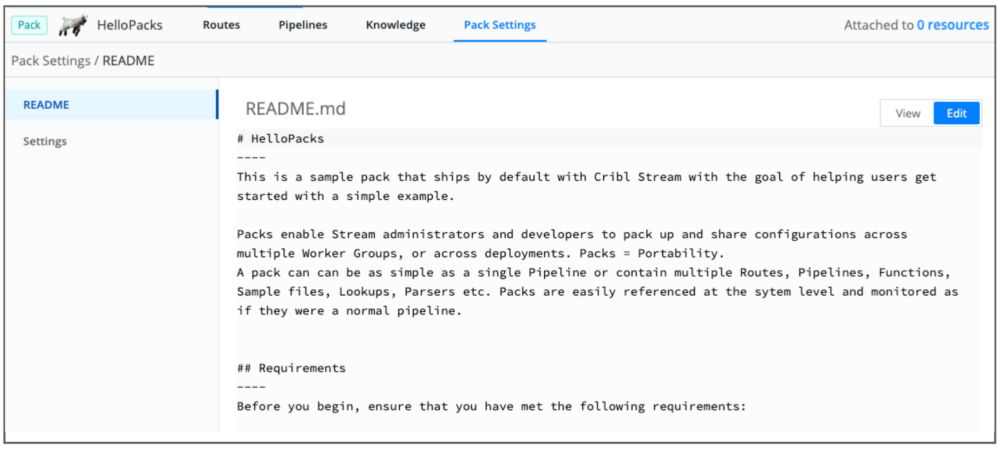
If you have a Cribl.Cloud account, you can also collaborate there by inviting team members to your Cribl.Cloud Organization. See Inviting and Managing Other Users.

Once your Pack is ready to share, we encourage you to submit it to the Cribl Packs Dispensary™ site. If you already have completed Packs on our GitHub repo, bring them over here!

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 **Pack Settings**. The left **README** tab will gain focus.
2. To populate the Pack's README file, toggle **View** to **Edit**, replace the placeholder markdown content, and **Save**.



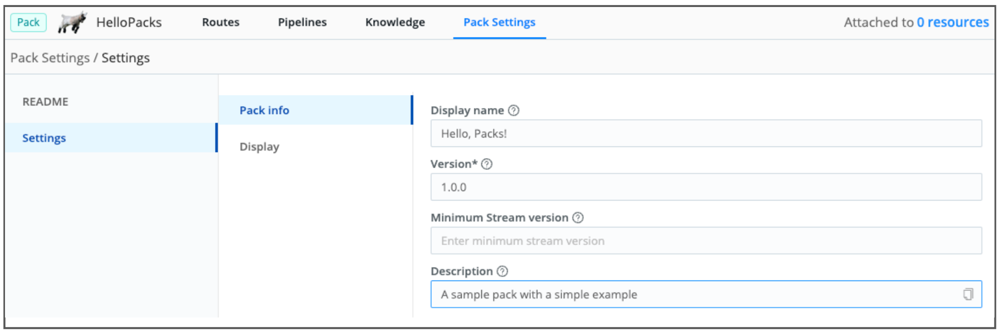
```
# HelloPacks
----
This is a sample pack that ships by default with Cribl Stream with the goal of helping users get
started with a simple example.

Packs enable Stream administrators and developers to pack up and share configurations across
multiple Worker Groups, or across deployments. Packs = Portability.
A pack can can be as simple as a single Pipeline or contain multiple Routes, Pipelines, Functions,
Sample files, Lookups, Parsers etc. Packs are easily referenced at the sytem level and monitored as
if they were a normal pipeline.


## Requirements
----
Before you begin, ensure that you have met the following requirements:
```

Editing Pack's README

3. To update other metadata, click the left **Settings** tab.



Display name: Hello, Packs!
Version*: 1.0.0
Minimum Stream version: Enter minimum stream version
Description: A sample pack with a simple example

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 Stream 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.

Copying a Pack

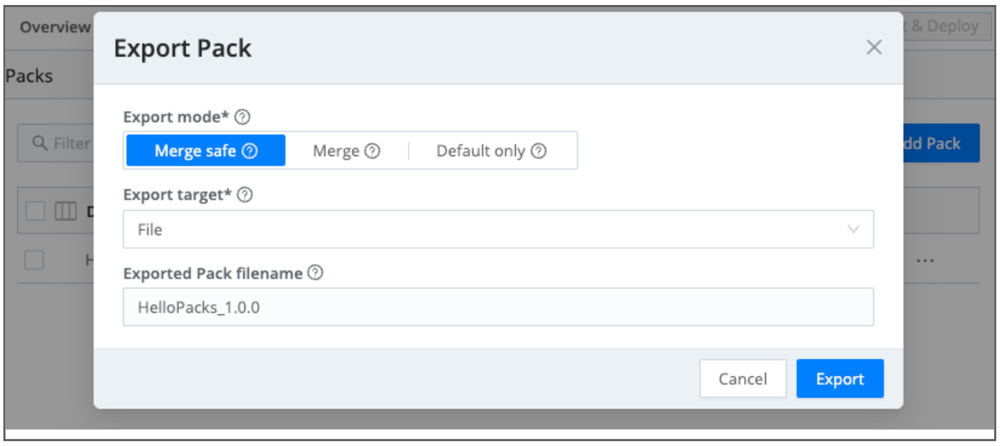
You can easily share Packs in a distributed deployment by copying them between Worker Groups.

1. You can copy one or more Packs.
   - To copy a single Pack: Select its menu icon on the Packs page and select **Copy to another Worker Group**.
   - To copy multiple Packs in one operation: Select their check boxes and then select **Copy selected Packs to another Worker Group**.
2. Select the Worker Groups to copy the Pack to.
3. Optionally, toggle **Overwrite** on to replace existing Packs with the same **Pack ID**.
4. Select **Copy**.

A status modal will provide a successful message or list any Packs that failed to copy.

Exporting a Pack

To export a newly created or modified Pack, click its menu icon on the Packs page and select **Export**.



Export mode*: Merge safe

Export target*: File

Exported Pack filename: HelloPacks_1.0.0

Exporting a Pack

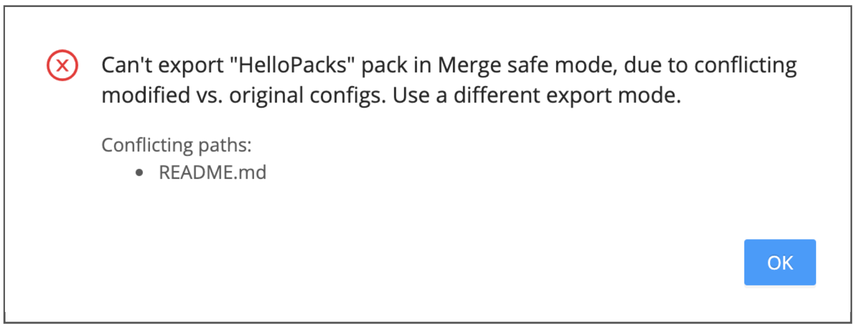
The resulting Export Pack modal provides the following options.

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.
- **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 Stream can't readily merge conflicting, modified contents with the Pack's original contents:



Can't export "HelloPacks" pack in Merge safe mode, due to conflicting modified vs. original configs. Use a different export mode.

Conflicting paths:
- README.md

Merge safe error

If you encounter an error like the example shown above, use the **Merge** or **Default only** export mode instead.

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 Stream 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 Worker Group 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 Worker Group 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` Worker Group, and use the `>` redirect to write the exported Pack to a file named `goat-herd.crbl`:

```
curl -X GET -H "Authorization: Bearer $token" 'https://stream:9000/api/v1/m/default
```

This request returns an octet-stream attachment which is downloaded as a `.crbl` file. And voilà, you have exported your Pack.

Install via API

Installing the exported Pack in a different Worker Group 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 Worker Group 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/group42
```

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 Worker Group.

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"
```

Copy via API

To bulk-copy Packs between Worker Groups/Fleets, adapt and run this **Clone pack** API call, referencing a source Worker Group, destination Worker Group(s)/Worker Group(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` Worker Groups:

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json"
```

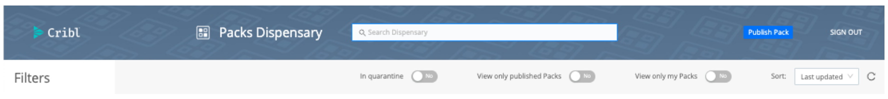
13.5.1. Packs Publication Standards

This page outlines the process for Cribl Community members to publish Cribl Stream Packs to the Cribl Packs Dispensary. It also lists standards that apply to all publicly available Community Packs.

Publication Overview

Publishing your Pack is a five-stage process:

1. **Prepare**
   - Prior to starting development, read the Pack Review Summary and Pack Review Checklist below. This will help you create a Pack that can easily pass the review process. Decide on the problems you're trying to solve, and determine the dataset(s), Sources, and Destinations you're going to work with.
2. **Develop your Pack**
   - Development is generally done in your own Cribl environment, using saved samples to build out pipelines. During this stage, you might consider working collaboratively within Cribl's Community Slack `#packs` channel. When the Pack is ready for review, export the `.crbl` file using the Cribl UI.
3. **Test your Pack**
   - Cribl strongly recommendeds that you test the Pack yourself, in your own environment. To test that all content was exported correctly, import your Pack into a fresh Cribl environment, such as a newly created Docker instance, or to a newly created Worker Group in your Cribl environment. Upload the Pack, then use the Pack Review Checklist to verify that the Pack will pass review.
4. **Submit your Pack for review**
   - Sign into, or create an account on, the Cribl Packs Dispensary site. You can use your Cribl.Cloud account's email address to create the Cribl Packs Dispensary account.
   - If you don't have a Cribl.Cloud account, Cribl automatically creates a new one for you when you create your Cribl Packs Dispensary account.
   - Once you sign in, you'll see the Dispensary screen with the **Publish Pack** button.



Packs Dispensary controls

   - Click **Publish Pack**, and then click **Upload Pack**.
   - Select the `.crbl` Pack file you want to publish, and click **Submit**. You will be prompted to agree to the terms of Cribl's Pack Developer Agreement. As a Pack author, you must electronically

acknowledge that you have read the Agreement, and that you intend to adhere to its requirements.

> Once the Pack has been uploaded, Cribl will review your Pack. The review process typically takes a week or more. Once your Pack is reviewed, you'll receive an email from `packs@cribl.io` either informing you that it was accepted, or providing the reason(s) why it was rejected.

5. **Once your accepted Pack is published to the Cribl Packs Dispensary, it's time to celebrate!**

Pack Review Summary

To ensure consistency across published Packs, Cribl evaluates all submitted Packs against the checklist below. Any Pack that fails to satisfy the checklist's requirements will be rejected.

Cribl recommends:

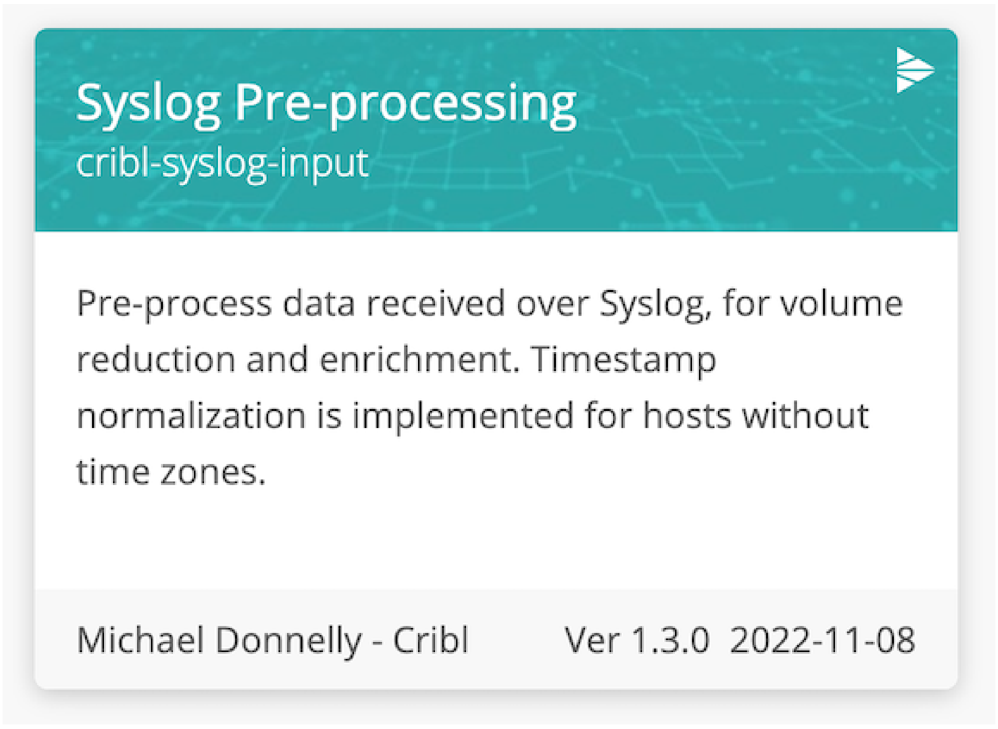
- Check your Pack against the checklist **before** trying to submit the Pack. This way you can avoid having your Pack rejected for something easily fixed.
- Install the **Syslog Pre-processing** (`cribl-syslog-input`) Pack as a reference – it includes clear examples of each item on the checklist.

Pack Review Checklist

1. Naming

Every Pack must have both a Pack ID and a Display Name.

When you view Packs in the Dispensary, the Display Name appears above the Pack ID. For example, **Syslog Pre-processing** is a Display name while `cribl-syslog-input` is a Pack ID:


Syslog Pre-processing
cribl-syslog-input

Pre-process data received over Syslog, for volume reduction and enrichment. Timestamp normalization is implemented for hosts without time zones.

Michael Donnelly - Cribl Ver 1.3.0 2022-11-08

Display Name and Pack ID

Pack ID

The Pack ID:

- Must start with either `cc-` (community-contributed) or `cribl-` (Cribl-contributed). Packs not satisfying this requirement will be immediately rejected.
- Should (by convention) be all lowercase, using the hyphen (`-`) as a delimiter.
- Should **not** include the word "pack."

For Packs normalizing to or from Elastic's Common Schema, the Pack ID (but not the Display Name) should include `-source` or `-dest`.

Display Name

The Display Name is a human-friendly version of the Pack name that:

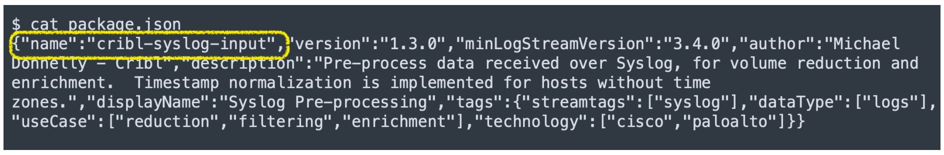
- Omits `cc-` and `cribl-`.
- Uses spaces where appropriate, and proper capitalization.
- If it includes a company or product name, matches the company's preferred spelling, spaces, and capitalization.

You can view or edit the Display Name in the Pack's **Settings** screen.

How to Review the Pack ID

This is the same procedure that the Cribl will perform when reviewing your Pack.

1. Extract the `.crbl` tarball that was submitted to the Dispensary:
   `tar zxvf <your_filename>.crbl`
2. View the `package.json` file:
   `cat package.json`
3. Check the value of the `name` field: this should be the Pack ID that you want.



```
$ cat package.json
{"name":"cribl-syslog-input","version":"1.3.0","minLogStreamVersion":"3.4.0","author":"Michael
Donnelly - Cribl","description":"Pre-process data received over Syslog, for volume reduction and
enrichment.  Timestamp normalization is implemented for hosts without time
zones.","displayName":"Syslog Pre-processing","tags":{"streamtags":["syslog"],"dataType":["logs"],
"useCase":["reduction","filtering","enrichment"],"technology":["cisco","paloalto"]}}
```

Verifying the Pack ID

2. Pack Settings

You must review your Pack's settings to verify that the Pack satisfies all of the following criteria.

Display Name

A human-friendly version of the Pack name as defined above.

Version

The version number in the submitted Pack must have a corresponding entry in the README Release Notes.

- The minimum version for initial publication is `0.1.0`. A Pack with a lower version **will be rejected**.
- A release that is considered "beta quality" should be designated as version `0.9.0`.

Description

Use this as a brief "hook" of one to two sentences. Example: `This pack for Syslog inputs will reduce volume, and address timestamp normalization for Syslog senders that omit timezones.`

Author

Community-contributed Packs may include the author’s name, a company name, or both.

All three of the following examples are valid:

1. `Jon Smith`
2. `Bazookatron`
3. `Jon Smith - Bazookatron`

Cribl-published Packs must use author’s full name, plus `- Cribl`. Example: `Michael Donnelly - Cribl`.

Data Type, Use Cases, Technologies

Select all that apply.

Tags

Tags are optional but recommended, especially where multiple Packs are intended to work together. Tag examples include: `OCSF`, `AWS`.

Logo

The logo is optional, but recommended. You can include a logo associated with the technology addressed by the Pack. For example, a Windows logo for Windows events, an AWS Cloudwatch logo for AWS Cloudwatch data, and so on.

You can set the logo via **Pack Settings** > **Settings** > **Display**.

3. Routes

A Pack submitted must include at least one Route in addition to the default Route.

- A Pack’s Routes page must have a Route, with an appropriate filter, routing to a Pipeline.
- The filter should match **only** the set of data that the Pipeline is designed to process. The filter must **not** be set to `true`.
- If the Pack deals with multiple datasets, each dataset should have its own Route/Pipeline pair. (For a good example, look at the PAN Pack.)
- The Pack must not use the default Route to send events to the Pack’s custom Pipeline. The default Route’s Pipeline may be set to `main`, `devnull`, or `passthru` to handle events that do not match the Pack’s filter(s).

- Each Route must include a short description of the dataset being matched.
- Pairs of Routes and Pipelines should have matching names.

4. Pipelines

A Pack submitted must include at least one Pipeline in addition to the stock Pipelines (`devnull`, `main`, `passthru`, `prometheus_metrics`, etc.).

- If the Pack deals with multiple datasets, there must be one Pipeline (and one Route) for each dataset. (For a good example, look at the PAN pack.)
- The Pack must not use the stock Pipelines. Leave these Pipelines unmodified.
- Every pipeline must begin with a Comment Function that gives an overview of the pipeline's operation as a whole. This Comment is a synopsis of what the entire Pipeline will be doing. Refer to the **Syslog Pre-processing** (`cribl-syslog-input`) Pack for an example of the expected level of detail.
- If Function Groups are used within the Pipeline, each Function Group must begin with a Comment Function that gives an overview of what will happen within that Function Group.
- Every Function must have a Description that briefly explains exactly what is happening in that particular Function. During review, change the Pipeline view option to display the Description for each Function.
- Using the sample files provided with the Pack, validate that the Functions work as detailed in the Comment at the top of the Pipeline.

5. Sample Files

The Pack must include at least one sample file for each of its Pipelines.

Sample files:

- Must **not** have an expiration date.
- Must **never** include PII, customer data, or customer hostnames.
- Must be named in such a way that it's obvious which sample file(s) apply to each Pipeline.
- Must contain samples that are adequate for a customer to use to review the operation of a Pipeline. To allow performance analysis of the corresponding Pipeline, a sample should include 20 or more events, with 100 or more recommended.
- May be reused across Pipelines, but it is preferable to include multiple data samples, each specific to a Pipeline's Route and filter.
- Must work as expected on a clean installation of the Pack. This applies to all samples listed in the Pack – Pack developers should test this before submitting the Pack.

6. README

The `README` is the documentation others will use in selecting and deploying the Pack. The `README` requirements ensure that others will be able to deploy the Pack successfully.

> The default sections for README, as created when you click **Create Pack**, do not match the required and optional sections in the checklist. You will need to delete or replace certain sections.

In the Pack's **Pack Settings** > **README** screen, validate that the README text is correct. The following rules apply to all sections of the README:

- Proper grammar will help the users of your Pack. Use a grammar and spelling checker to make sure the README is clear and readable.
- If the README includes any text that looks like a placeholder, the Pack will be rejected.
- Trademark or product names must be spelled and capitalized exactly as the company that owns them would dictate. This requirement also applies to Pack settings, inline comments, etc. Examples: Amazon (not amazon), GCP (not Gcp), CrowdStrike (not Crowdstrike).
- The README must include all of the sections listed below (sections **6a** through **6f**). The README may include additional sections as necessary.

6a. About this Pack

Think of this initial section as the first paragraph of an essay, or as the only paragraph a person will read when deciding to use the Pack. It must clearly state what the Pack does. Start with the benefits of using the Pack. In sum, this section should answer the question, **Why should anyone use this Pack?**

If the Pack is tied to a specific data source, dataset, or destination, spell that out clearly here – even if that source or destination is in the Pack name.

6b. Deployment

This **required** section must include all instructions necessary to get the Pack up and running. If the Pack requires any configuration after initial installation - but before it's put it into production - you must specify the configuration steps here.

Filters: If applicable, say what filter must be used to route data to the Pack, or explain how to configure the filter. A filter based on `__inputId` is much more efficient than a match against `_raw`.

Sources: If the Pack requires a specific Source, include information on how to configure that Source. At a minimum, this needs to cover setting up the Source within Cribl. In some cases, this might include information on how to set up an external environment for collection by Cribl as well.

Destinations: If the Pack requires a specific Destination, include information on how to configure that Destination. Does it need a post-processing Pipeline tied to the Pack?

Pipelines: If the Pack includes Functions (within a Pipeline) that need to tailored, enabled, or disabled, include a full explanation and instructions.

Lookups: If the Pack includes Lookup files that need to be tailored, you must specify the configuration steps here.

6c. Upgrades

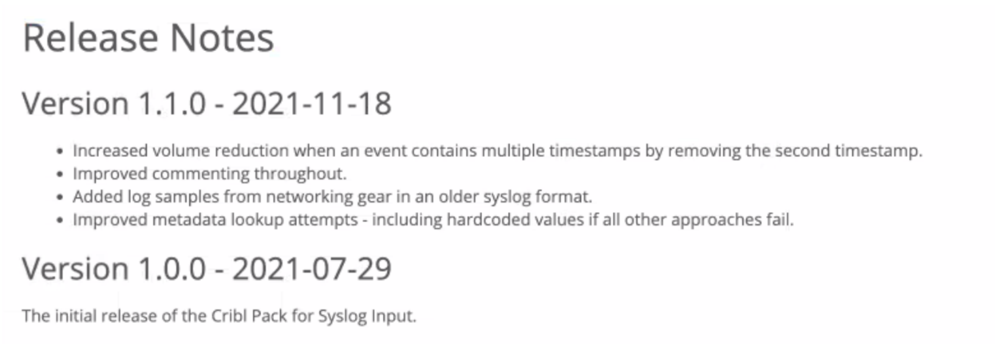
This **optional** section is strongly recommended if you expect others to customize their copy of the Pack and later upgrade the modified Pack.

- If included, this section should cover the usual set of problems with Pack upgrades as described in Upgrading an Existing Pack, and should explain the specifics of upgrading this particular Pack.

6d. Release Notes

This **required** section is a list of release notes where each release note:

- Starts with the version number and date of release, separated by a space, a hyphen, and a space. For example: `Version 1.2.0 - 2022-07-11`.
- Continues with bullet points that describe what has changed and/or is new in the release.


Release Notes

Version 1.1.0 - 2021-11-18

- Increased volume reduction when an event contains multiple timestamps by removing the second timestamp.
- Improved commenting throughout.
- Added log samples from networking gear in an older syslog format.
- Improved metadata lookup attempts - including hardcoded values if all other approaches fail.

Version 1.0.0 - 2021-07-29

The initial release of the Cribl Pack for Syslog Input.

Release notes

Release notes must adhere to the following rules:

- Designate the initial release version `Initial release`.
- The most current version listed must exactly match the Pack version in **Pack Settings** > **Settings**.

- Release notes versions must be listed starting with the most recent (in descending chronological order).

6e. Contributing to the Pack

This **required** section explains how to contribute to the Pack.

- The information provided must be **actionable**. A section reading `[Contributor instructions go here.]` would be rejected.
- Cribl recommends that you refer potential contributors to Cribl's Community Slack. For example, Cribl-created Packs read as follows:

```
To contribute to the Pack, please connect with us on [Cribl Community Slack](h
```

6f. License

This **required** section must include text that designates **and links to** a legitimate license.

- Community-contributed Packs must include a license that allows for use by the general public, such as Apache 2.0, MIT, or GNU.
- Built-by-Cribl Packs must use the following snippet:

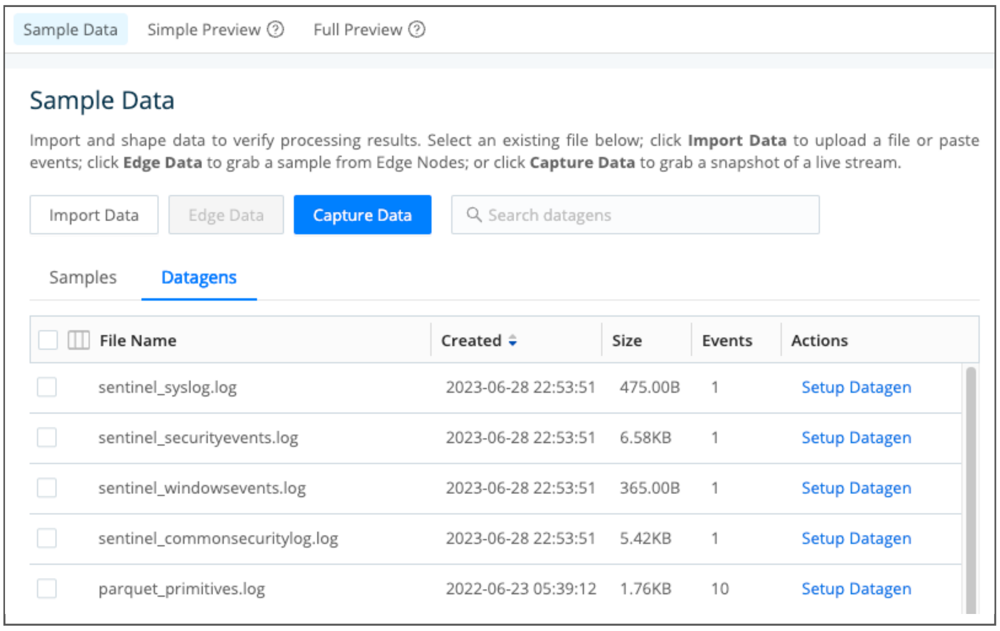
```
This Pack uses the following license: [Apache 2.0](https://github.com/criblio/
```

13.6. Using Datagens

Data generators for testing and troubleshooting

Cribl Stream'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.



Sample Data

Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a snapshot of a live stream.

File Name	Created	Size	Events	Actions
sentinel_syslog.log	2023-06-28 22:53:51	475.00B	1	Setup Datagen
sentinel_securityevents.log	2023-06-28 22:53:51	6.58KB	1	Setup Datagen
sentinel_windowsevents.log	2023-06-28 22:53:51	365.00B	1	Setup Datagen
sentinel_commonsecuritylog.log	2023-06-28 22:53:51	5.42KB	1	Setup Datagen
parquet_primitives.log	2022-06-23 05:39:12	1.76KB	10	Setup Datagen

Preview pane – add samples via paste, attach/upload file, or live capture

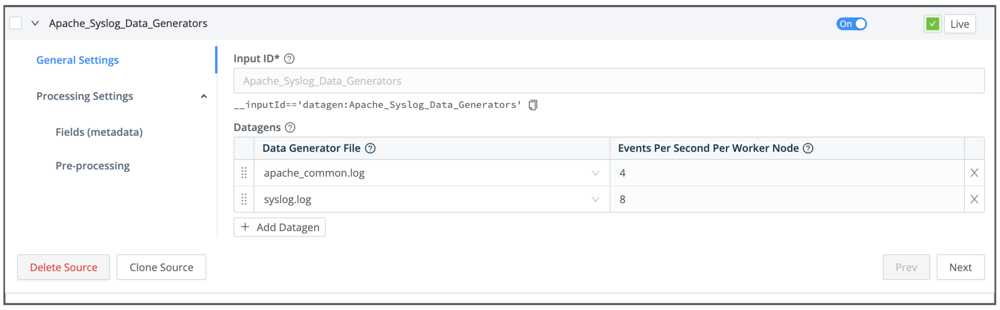
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen Source to use the template to generate real-time data at a given EPS (events per second) rate.

Enabling a Datagen

To see how datagens work, start by enabling a pair of Cribl Stream'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**.



Apache_Syslog_Data_Generators — On — Live

General Settings · Processing Settings · Fields (metadata) · Pre-processing

Input ID*: Apache_Syslog_Data_Generators

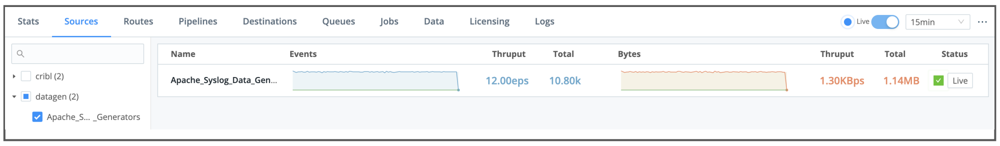
`__inputId=='datagen:Apache_Syslog_Data_Generators'`

Datagens

Data Generator File	Events Per Second Per Worker Node
apache_common.log	4
syslog.log	8

Selecting datagens files and event rates

On the **Monitoring** page, under **Sources**, search for `datagen` and confirm that the Source is generating data.



Name	Events	Thruput	Total	Bytes	Thruput	Total	Status
Apache_Syslog_Data_Gen...		12.00eps	10.80k		1.30KBps	1.14MB	✓ Live

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:

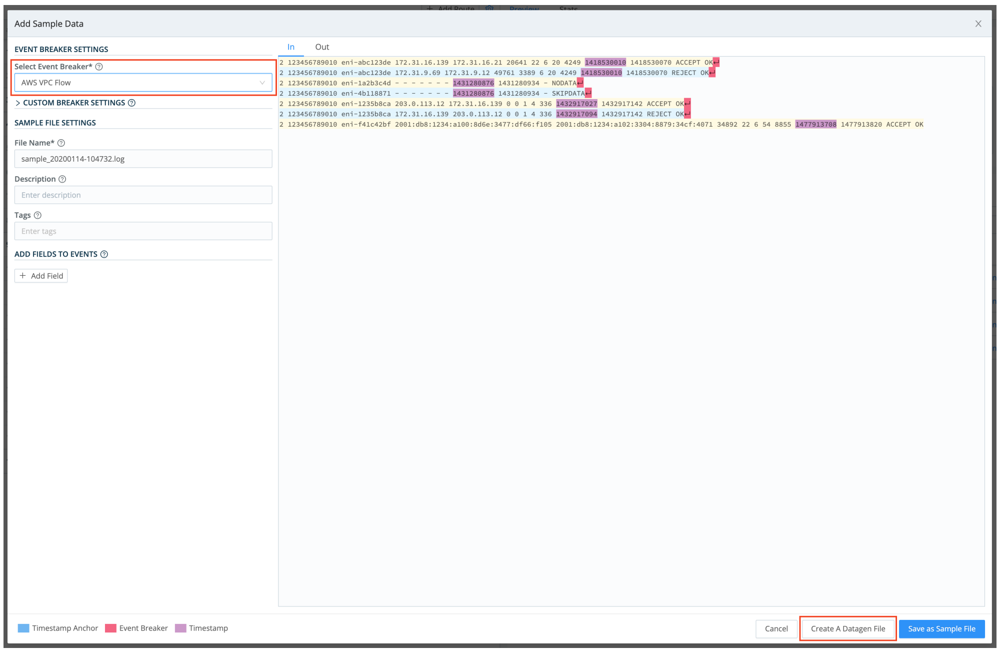
Sample VPC Flow Logs

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 141853001
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 141853001
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 14329
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 14329
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:1234:a1
```

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**.


Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: AWS VPC Flow

CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*: sample_20200114-104732.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

In | Out

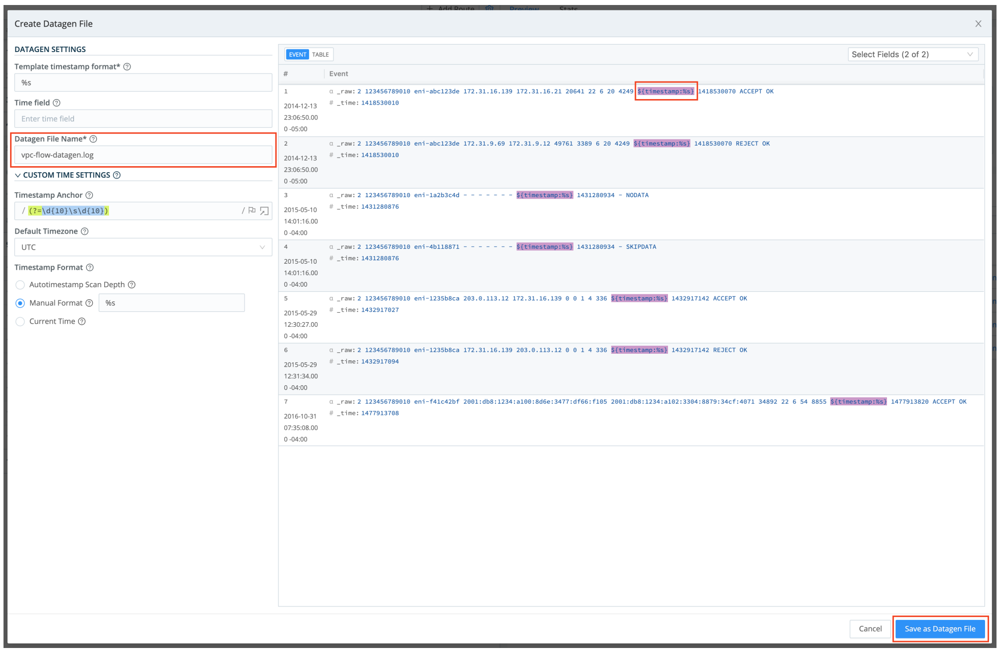
```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010 1418530070 ACCEPT OK
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418530010 1418530070 REJECT OK
2 123456789010 eni-1a2b3c4d - - - - - - - 1431280876 1431280934 - NODATA
2 123456789010 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
```

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**.


Create Datagen File

DATAGEN SETTINGS

Template timestamp format*: %s

Time field: Enter time field

Datagen File Name*: vpc-flow-datagen.log

CUSTOM TIME SETTINGS

Timestamp Anchor: / (?=\d{10}\s\d{10}) /

Default Timezone: UTC

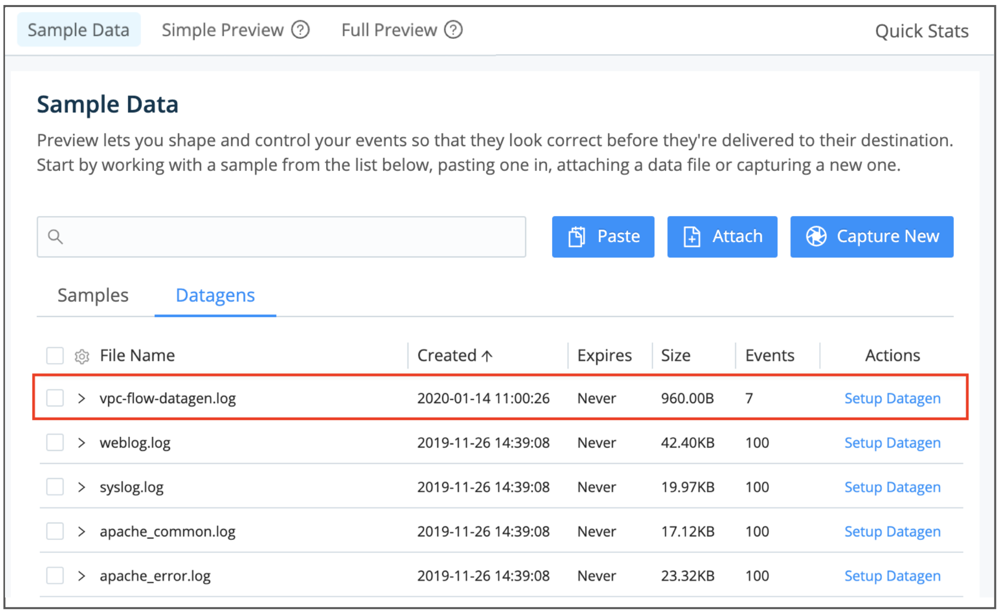
Timestamp Format: Autotimestamp Scan Depth | [✓ Manual Format] %s | Current Time

EVENT | TABLE — Select Fields (2 of 2)

```
#  Event
1  _raw: 2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 ${timestamp:%s} 1418530070 ACCEPT OK
   _time: 1418530010
2014-12-13 23:06:50.000 -05:00
2  _raw: 2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 ${timestamp:%s} 1418530070 REJECT OK
   _time: 1418530010
2014-12-13 23:06:50.000 -05:00
3  _raw: 2 123456789010 eni-1a2b3c4d - - - - - - - ${timestamp:%s} 1431280934 - NODATA
   _time: 1431280876
2015-05-10 14:01:16.000 -04:00
4  _raw: 2 123456789010 eni-4b118871 - - - - - - - ${timestamp:%s} 1431280934 - SKIPDATA
   _time: 1431280876
2015-05-10 14:01:16.000 -04:00
5  _raw: 2 123456789010 eni-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 ${timestamp:%s} 1432917142 ACCEPT OK
   _time: 1432917027
2015-05-29 12:30:27.000 -04:00
6  _raw: 2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 ${timestamp:%s} 1432917142 REJECT OK
   _time: 1432917094
2015-05-29 12:31:34.000 -04:00
7  _raw: 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 ${timestamp:%s} 1477913820 ACCEPT OK
   _time: 1477913708
2016-10-31 07:35:08.000 -04:00
```

Saving a named datagen template

To confirm that the datagen file has been created, check **Preview** > **Datagens**.



Sample Data | Simple Preview | Full Preview | Quick Stats

Sample Data

Preview lets you shape and control your events so that they look correct before they're delivered to their destination. Start by working with a sample from the list below, pasting one in, attaching a data file or capturing a new one.

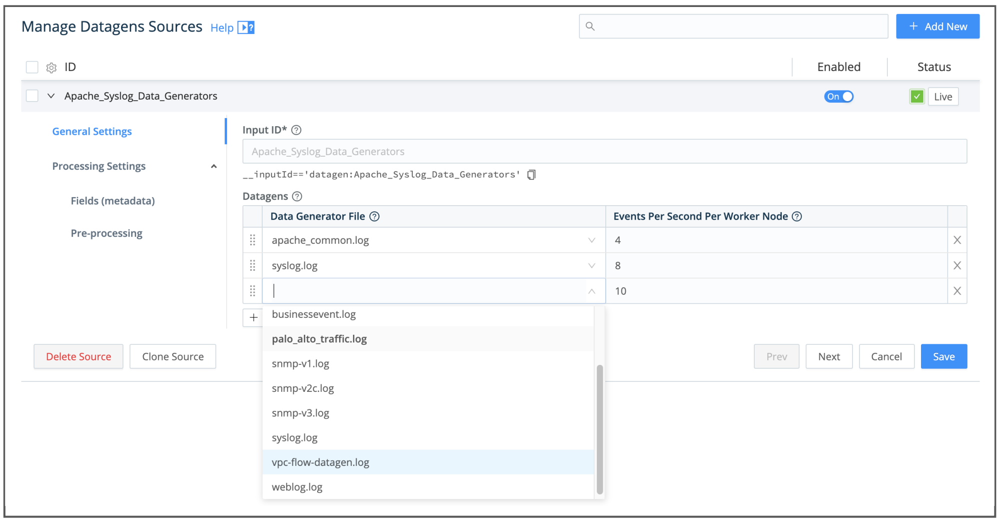
Paste | Attach | Capture New

Samples | Datagens

File Name	Created ↑	Expires	Size	Events	Actions
vpc-flow-datagen.log	2020-01-14 11:00:26	Never	960.00B	7	Setup Datagen
weblog.log	2019-11-26 14:39:08	Never	42.40KB	100	Setup Datagen
syslog.log	2019-11-26 14:39:08	Never	19.97KB	100	Setup Datagen
apache_common.log	2019-11-26 14:39:08	Never	17.12KB	100	Setup Datagen
apache_error.log	2019-11-26 14:39:08	Never	23.32KB	100	Setup Datagen

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.



Manage Datagens Sources Help | + Add New

ID	Enabled	Status
Apache_Syslog_Data_Generators	On	✓ Live

General Settings | Processing Settings | Fields (metadata) | Pre-processing

Input ID*: Apache_Syslog_Data_Generators

```
__inputId=='datagen:Apache_Syslog_Data_Generators'
```

Datagens

Data Generator File	Events Per Second Per Worker Node
apache_common.log	4
syslog.log	8
	10

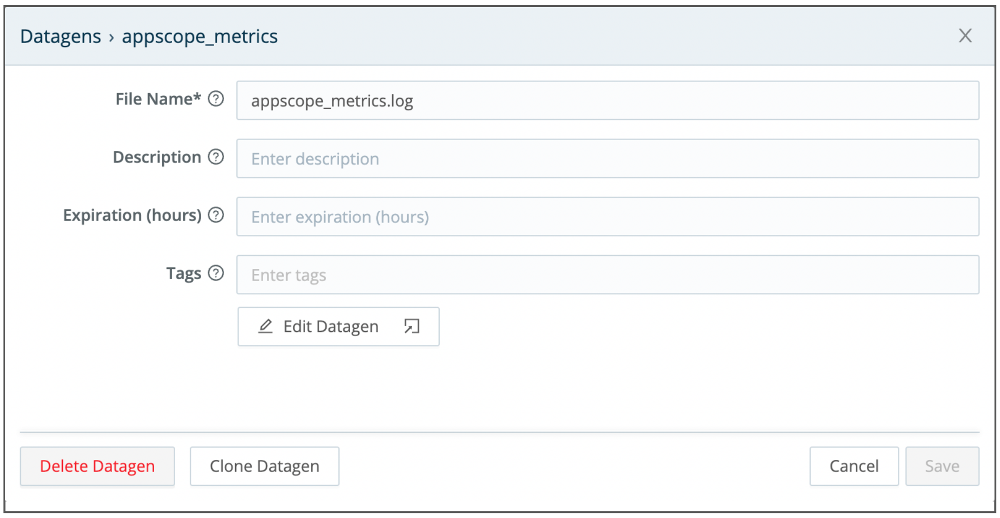
businessevent.log
palo_alto_traffic.log
snmp-v1.log
snmp-v2c.log
snmp-v3.log
syslog.log
vpc-flow-datagen.log
weblog.log

Delete Source | Clone Source | Prev | Next | Cancel | Save

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**).

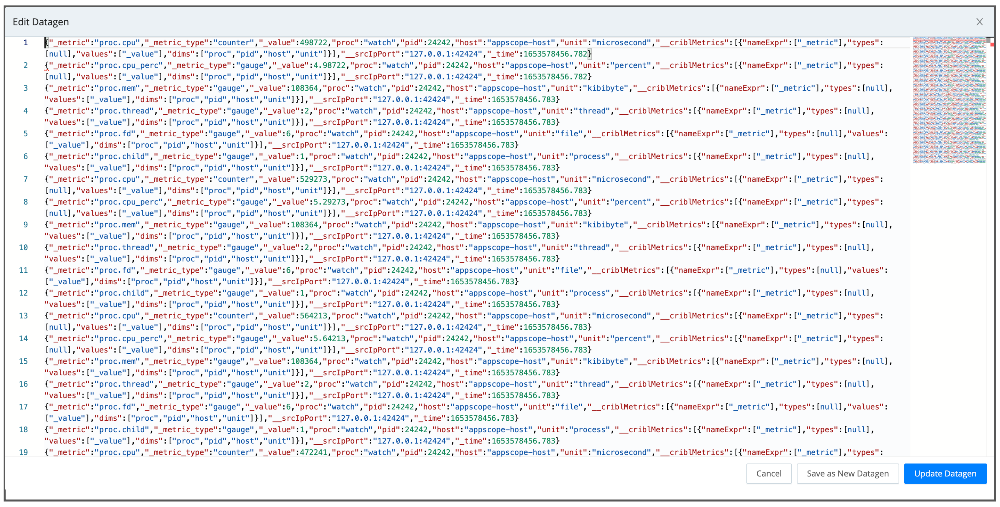


Datagens › appscope_metrics

File Name*	appscope_metrics.log
Description	Enter description
Expiration (hours)	Enter expiration (hours)
Tags	Enter 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.



Edit Datagen

```
{"_metric":"proc.cpu","_metric_type":"counter","_value":498722,"proc":"watch","pid":24242,"host":"appscope-host","unit":"microsecond","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.782}
{"_metric":"proc.cpu_perc","_metric_type":"gauge","_value":4.98722,"proc":"watch","pid":24242,"host":"appscope-host","unit":"percent","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.782}
{"_metric":"proc.mem","_metric_type":"gauge","_value":108364,"proc":"watch","pid":24242,"host":"appscope-host","unit":"kibibyte","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.thread","_metric_type":"gauge","_value":2,"proc":"watch","pid":24242,"host":"appscope-host","unit":"thread","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.fd","_metric_type":"gauge","_value":6,"proc":"watch","pid":24242,"host":"appscope-host","unit":"file","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.child","_metric_type":"gauge","_value":1,"proc":"watch","pid":24242,"host":"appscope-host","unit":"process","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.cpu","_metric_type":"counter","_value":529273,"proc":"watch","pid":24242,"host":"appscope-host","unit":"microsecond","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.cpu_perc","_metric_type":"gauge","_value":5.29273,"proc":"watch","pid":24242,"host":"appscope-host","unit":"percent","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.mem","_metric_type":"gauge","_value":108364,"proc":"watch","pid":24242,"host":"appscope-host","unit":"kibibyte","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.thread","_metric_type":"gauge","_value":2,"proc":"watch","pid":24242,"host":"appscope-host","unit":"thread","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.fd","_metric_type":"gauge","_value":6,"proc":"watch","pid":24242,"host":"appscope-host","unit":"file","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.child","_metric_type":"gauge","_value":1,"proc":"watch","pid":24242,"host":"appscope-host","unit":"process","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.cpu","_metric_type":"counter","_value":564213,"proc":"watch","pid":24242,"host":"appscope-host","unit":"microsecond","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.cpu_perc","_metric_type":"gauge","_value":5.64213,"proc":"watch","pid":24242,"host":"appscope-host","unit":"percent","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.mem","_metric_type":"gauge","_value":108364,"proc":"watch","pid":24242,"host":"appscope-host","unit":"kibibyte","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.thread","_metric_type":"gauge","_value":2,"proc":"watch","pid":24242,"host":"appscope-host","unit":"thread","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.fd","_metric_type":"gauge","_value":6,"proc":"watch","pid":24242,"host":"appscope-host","unit":"file","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.child","_metric_type":"gauge","_value":1,"proc":"watch","pid":24242,"host":"appscope-host","unit":"process","__criblMetrics":[{"nameExpr":["_metric"],"types":[null],"values":["_value"],"dims":["proc","pid","host","unit"]}],"__srcIpPort":"127.0.0.1:42424","_time":1653578456.783}
{"_metric":"proc.cpu","_metric_type":"counter","_value":472241,"proc":"watch","pid":24242,"host":"appscope-host","unit":"microsecond","__criblMetrics":[{"nameExpr":["_metric"],"types":
```

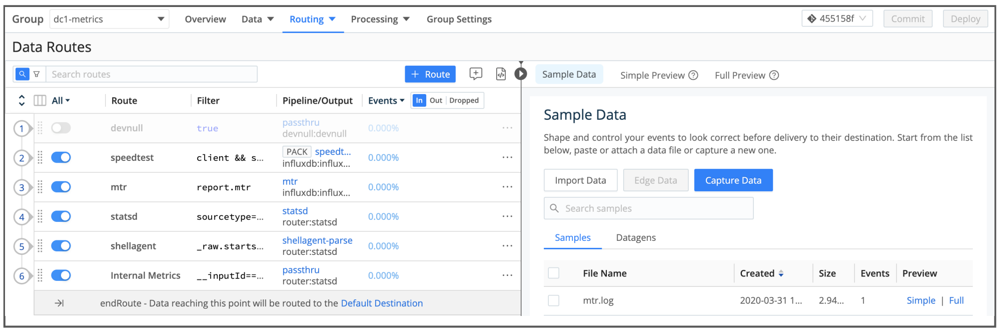
Editing a datagen

13.7. Data Preview

Cribl Stream'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.



Data Routes

	All	Route	Filter	Pipeline/Output	Events (In / Out / Dropped)
1		devnull	`true`	passthru devnull:devnull	0.000%
2		speedtest	`client && s...`	PACK speedt... influxdb:influx...	0.000%
3		mtr	`report.mtr`	mtr influxdb:influx...	0.000%
4		statsd	`sourcetype=...`	statsd router:statsd	0.000%
5		shellagent	`_raw.starts...`	shellagent-parse router:statsd	0.000%
6		Internal Metrics	`__inputId==...`	passthru router:statsd	0.000%

endRoute - Data reaching this point will be routed to the Default Destination

Sample Data

Shape and control your events to look correct before delivery to their destination. Start from the list below, paste or attach a data file or capture a new one.

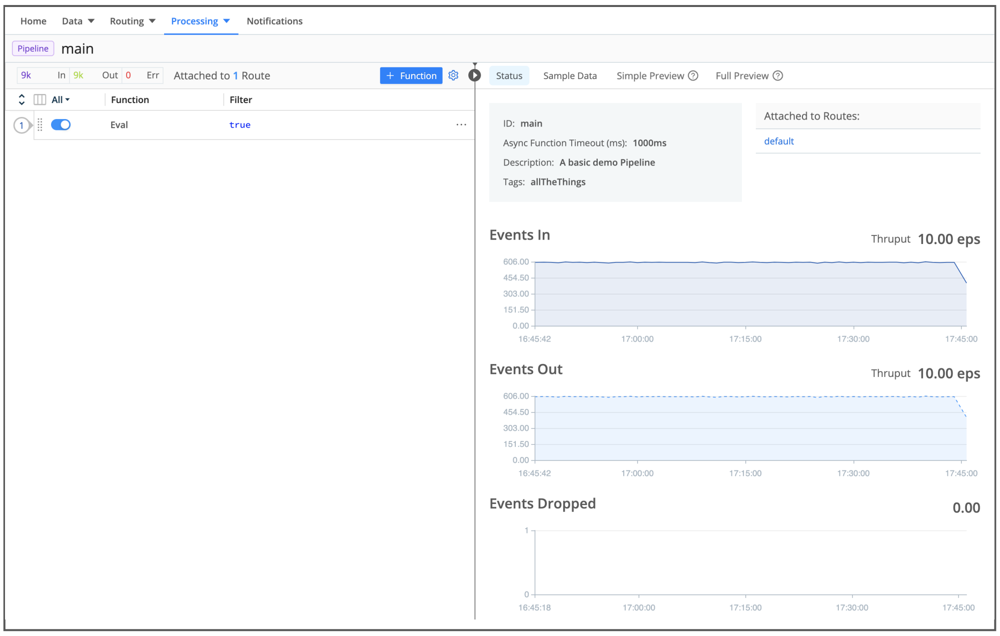
File Name	Created	Size	Events	Preview
mtr.log	2020-03-31 1...	2.94...	1	Simple | Full

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).



ID: main
Async Function Timeout (ms): 1000ms
Description: A basic demo Pipeline
Tags: allTheThings

Attached to Routes:
default

Pipeline status tab

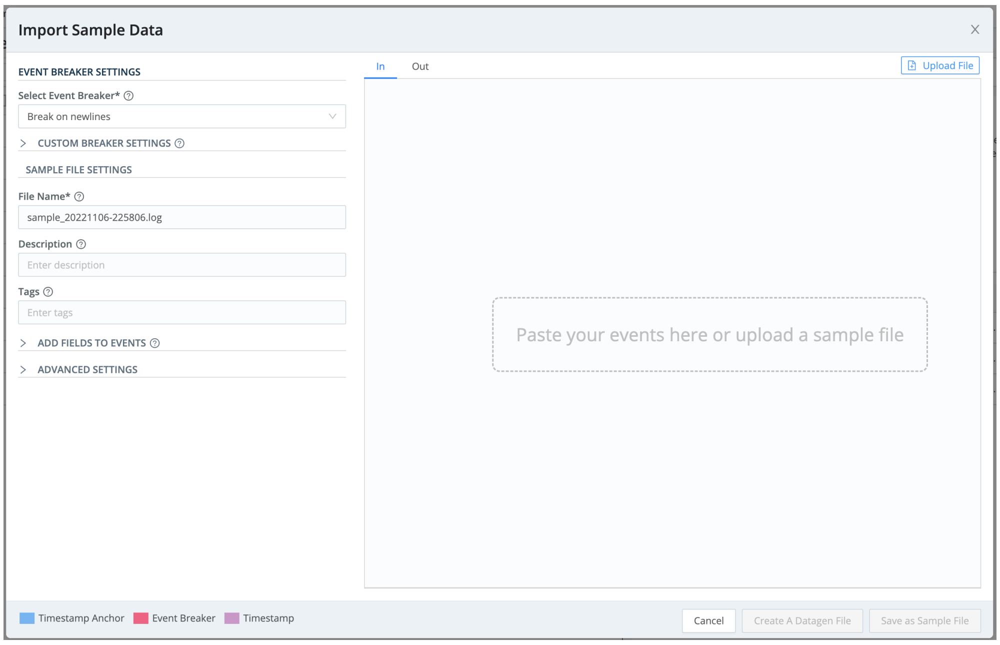
Adding Sample Data

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

EVENT BREAKER SETTINGS

Select Event Breaker*: Break on newlines

CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*: sample_20221106-225806.log

Description: Enter description

Tags: Enter tags

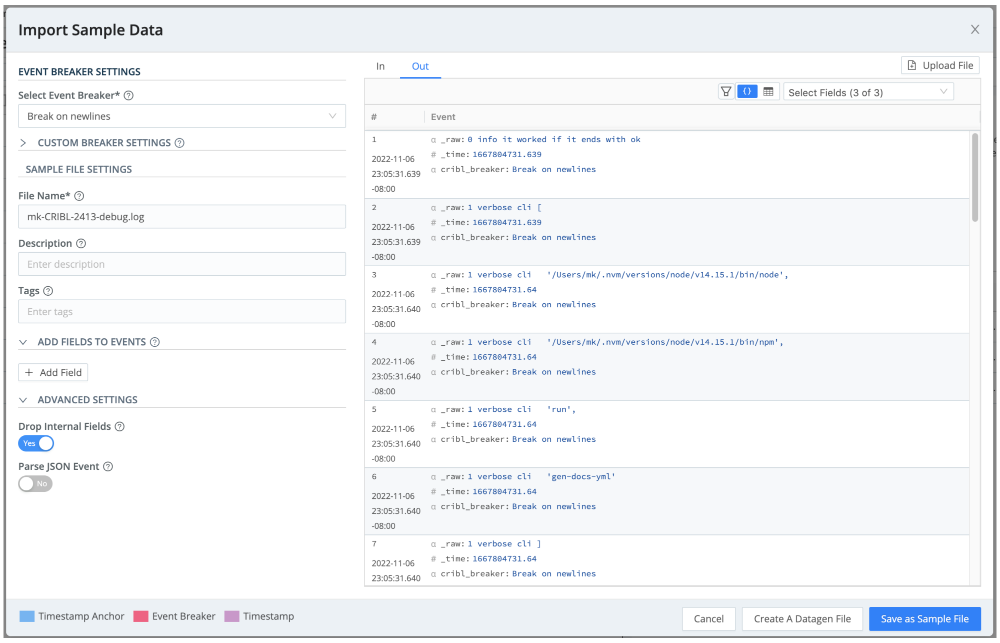
ADD FIELDS TO EVENTS

ADVANCED SETTINGS

Paste your events here or upload a sample 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.



Import Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: Break on newlines

CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*: mk-CRIBL-2413-debug.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

ADVANCED SETTINGS

Drop Internal Fields: Yes

Parse JSON Event: No

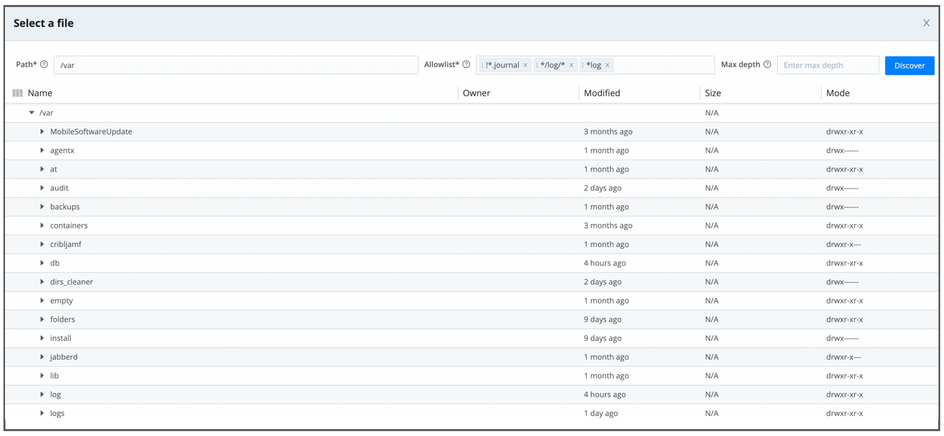
```
#   Event
1   _raw: 0 info it worked if it ends with ok
    _time: 1667804731.639
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.639 -08:00
2   _raw: 1 verbose cli [
    _time: 1667804731.639
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.639 -08:00
3   _raw: 1 verbose cli   '/Users/mk/.nvm/versions/node/v14.15.1/bin/node',
    _time: 1667804731.64
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.640 -08:00
4   _raw: 1 verbose cli   '/Users/mk/.nvm/versions/node/v14.15.1/bin/npm',
    _time: 1667804731.64
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.640 -08:00
5   _raw: 1 verbose cli   'run',
    _time: 1667804731.64
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.640 -08:00
6   _raw: 1 verbose cli   'gen-docs-yml'
    _time: 1667804731.64
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.640 -08:00
7   _raw: 1 verbose cli ]
    _time: 1667804731.64
    cribl_breaker: Break on newlines
2022-11-06 23:05:31.640
```

Working with imported data

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

Path* /var

Allowlist* !*.journal */log/* *log

Max depth Enter max depth

Name	Owner	Modified	Size	Mode
/var			N/A	
MobileSoftwareUpdate		3 months ago	N/A	drwxr-xr-x
agentx		1 month ago	N/A	drwx------
at		1 month ago	N/A	drwxr-xr-x
audit		2 days ago	N/A	drwx------
backups		1 month ago	N/A	drwx------
containers		3 months ago	N/A	drwxr-xr-x
cribljamf		1 month ago	N/A	drwxr-x---
db		4 hours ago	N/A	drwxr-xr-x
dirs_cleaner		2 days ago	N/A	drwx------
empty		1 month ago	N/A	drwxr-xr-x
folders		9 days ago	N/A	drwxr-xr-x
install		9 days ago	N/A	drwx------
jabberd		1 month ago	N/A	drwxr-x---
lib		1 month ago	N/A	drwxr-xr-x
log		4 hours ago	N/A	drwxr-xr-x
logs		1 day ago	N/A	drwxr-xr-x

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 Stream how to break the file or pasted content into events. Breaking will occur at the **start** of the match. Cribl Stream 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

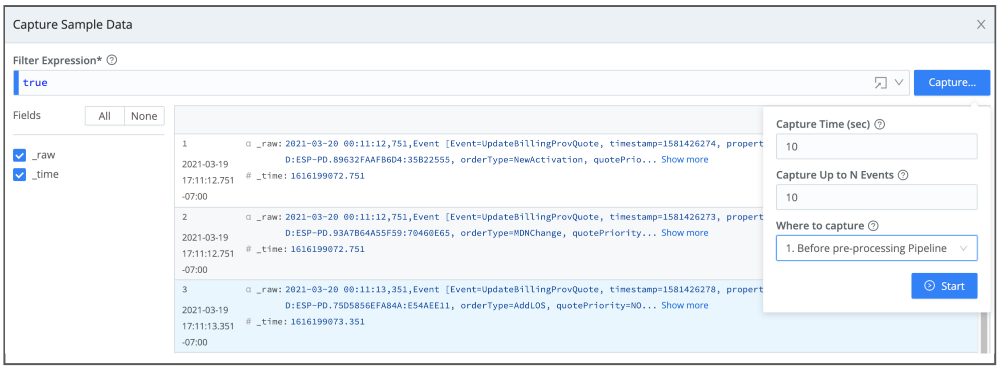
If you notice fragmented events, check whether Cribl Stream has added a `__timeoutFlush` internal field to them. This diagnostic field's presence indicates that the events were flushed because the Event Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes.

Capturing Sample Data

The **Capture Data** button opens a slightly different modal – it does not require event breaking.

In order to capture live data, you must have Worker Nodes registered to the Worker Group for which you're viewing events. You can view registered Worker Nodes from the **Status** tab in the Source.

In the composite screenshot below, we've already captured some events using the **Capture** drop-down.



Capture Sample Data

Filter Expression*: `true`

Fields: ✓ _raw, ✓ _time

```
1                  α _raw: 2021-03-20 00:11:12,751,Event [Event=UpdateBillingProvQuote, timestamp=1581426274, propert
2021-03-19                 D:ESP-PD.89632FAAFB6D4:35B22555, orderType=NewActivation, quotePrio... Show more
17:11:12.751       # _time: 1616199072.751
-07:00
2                  α _raw: 2021-03-20 00:11:12,751,Event [Event=UpdateBillingProvQuote, timestamp=1581426273, propert
2021-03-19                 D:ESP-PD.93A7B64A55F59:70460E65, orderType=MDNChange, quotePriority... Show more
17:11:12.751       # _time: 1616199072.751
-07:00
3                  α _raw: 2021-03-20 00:11:13,351,Event [Event=UpdateBillingProvQuote, timestamp=1581426278, propert
2021-03-19                 D:ESP-PD.75D5856EFA84A:E54AEE11, orderType=AddLOS, quotePriority=NO... Show more
17:11:13.351       # _time: 1616199073.351
-07:00
```

Capture Time (sec): 10

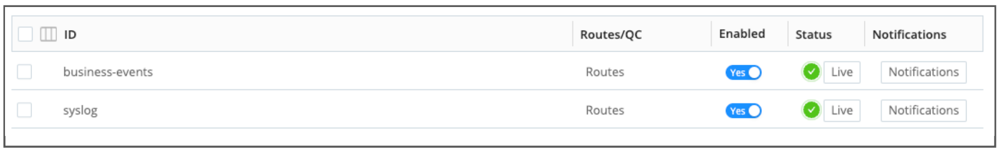
Capture Up to N Events: 10

Where to capture: 1. Before pre-processing Pipeline

Capture Data > Capture Sample Data modal

Capturing from a Single Source or Destination

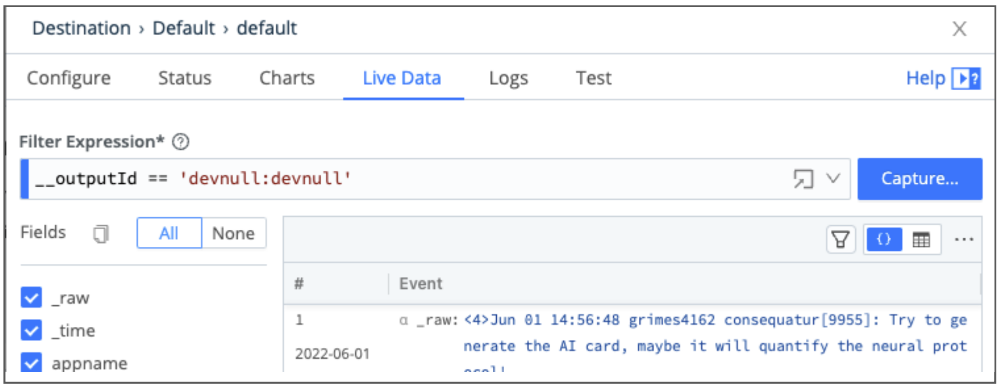
To capture data from a single enabled Source or Destination, it's fastest to use the Sources or Destinations UI instead of the Preview pane. You can initiate an immediate capture by clicking the **Live** button on the Source's or Destination's configuration row.



ID	Routes/QC	Enabled	Status	Notifications
business-events	Routes	Yes	✓ Live	Notifications
syslog	Routes	Yes	✓ Live	Notifications

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.



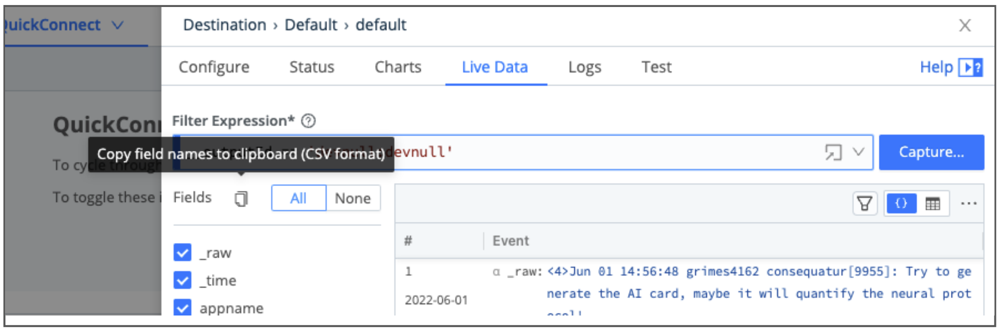
Filter Expression*: `__outputId == 'devnull:devnull'`

Fields: ✓ _raw, ✓ _time, ✓ appname

```
#    Event
1           α _raw: <4>Jun 01 14:56:48 grimes4162 consequatur[9955]: Try to ge
2022-06-01          nerate the AI card, maybe it will quantify the neural prot
```

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.



```
1           _raw: <4>Jun 01 14:56:48 grimes4162 consequatur[9955]: Try to ge
2022-06-01        nerate the AI card, maybe it will quantify the neural prot
```

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 a single instance at **Settings** > **General Settings > Limits** > **Storage** > **Max sample size**.
- On distributed Worker Groups at **Manage** > `<group-name>` > **Group Settings** > **General Settings** > **Limits** > **Storage Max sample size**.

In each location, the default limit is `256 KB`. You can adjust this upward (to a maximum of `3 MB`) or downward.

Very Large Integer Values

Cribl Stream'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:

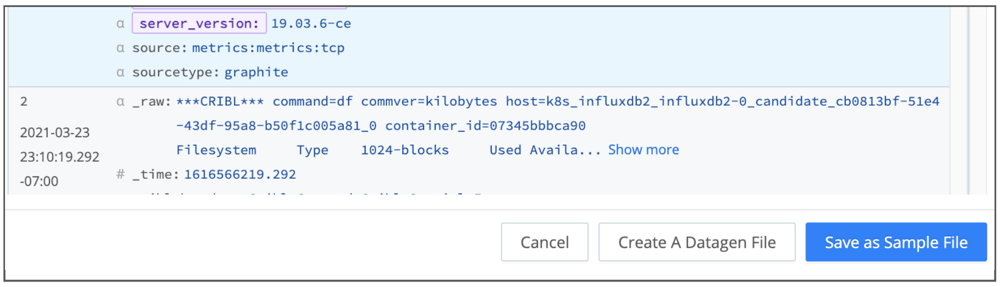
Symbol	Meaning
α	string
#	numeric
b	boolean
m	metric
{}	JSON object
[]	array

On JSON objects and arrays, you'll also see:

Symbol	Meaning
+	expandable
-	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.

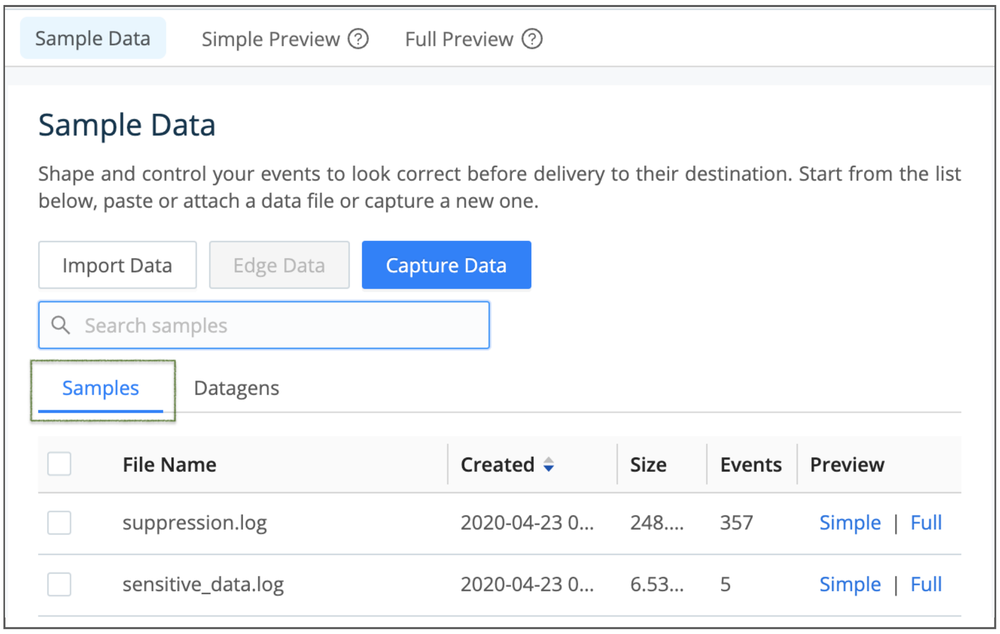


```
              α server_version: 19.03.6-ce
              α source: metrics:metrics:tcp
              α sourcetype: graphite
2             α _raw: ***CRIBL*** command=df commver=kilobytes host=k8s_influxdb2_influxdb2-0_candidate_cb0813bf-51e4
2021-03-23          -43df-95a8-b50f1c005a81_0 container_id=07345bbbca90
23:10:19.292        Filesystem     Type    1024-blocks     Used Availa... Show more
-07:00        # _time: 1616566219.292
```

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.



Sample Data

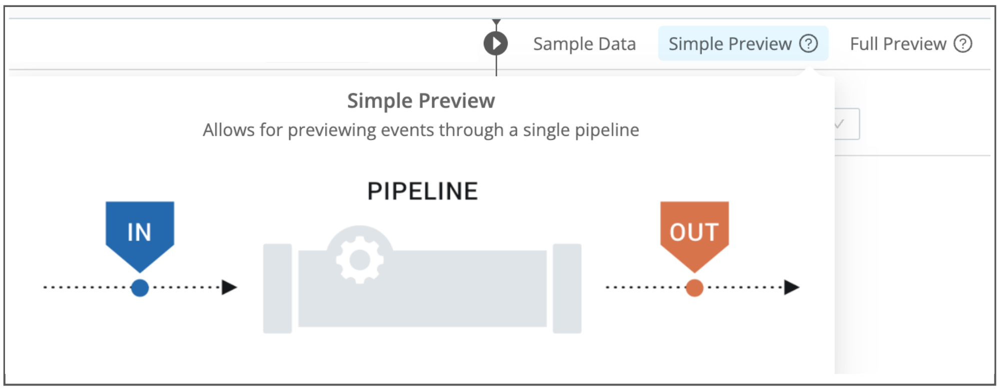
Shape and control your events to look correct before delivery to their destination. Start from the list below, paste or attach a data file or capture a new one.

	File Name	Created	Size	Events	Preview
	suppression.log	2020-04-23 0...	248....	357	Simple | Full
	sensitive_data.log	2020-04-23 0...	6.53...	5	Simple | Full

Managing sample files

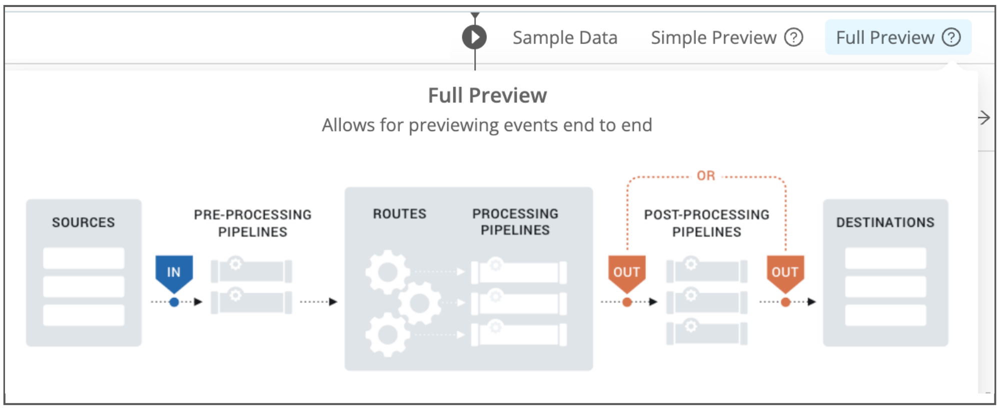
Simple Versus Full Preview

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

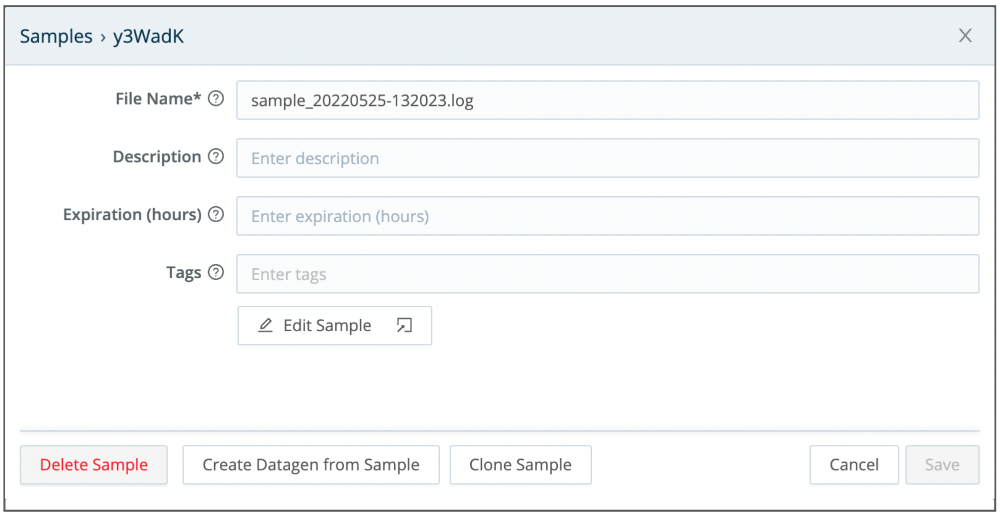
The **Full Preview** option gives you a choice of viewing events on the **OUT** side of either the processing or post-processing Pipeline. Selecting this option expands the Preview pane's upper controls to include an **Exit Point** drop-down, where you make this choice.



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**).

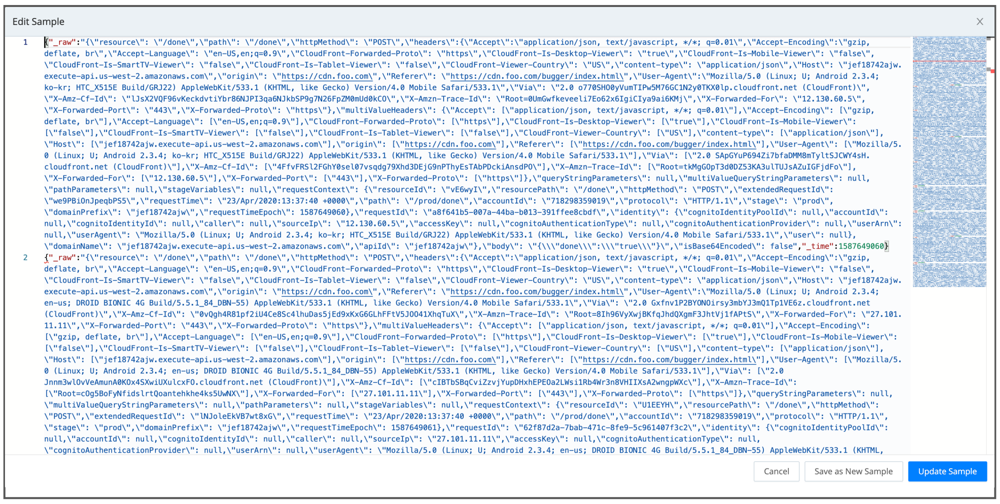


Samples › y3WadK

File Name*	sample_20220525-132023.log
Description	Enter description
Expiration (hours)	Enter expiration (hours)
Tags	Enter 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.

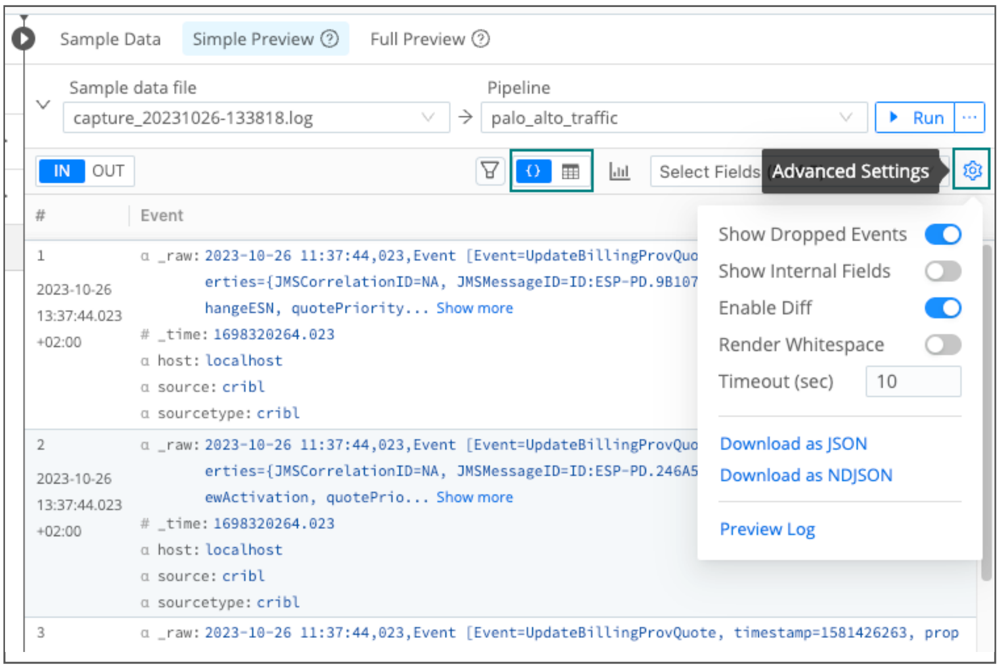


```
{"_raw":"{\"resource\": \"/done\",\"path\": \"/done\",\"httpMethod\": \"POST\",\"headers\":{\"Accept\":\"application/json, text/javascript, */*; q=0.01\",\"Accept-Encoding\":\"gzip,
deflate, br\",\"Accept-Language\": \"en-US,en;q=0.9\",\"CloudFront-Forwarded-Proto\": \"https\",\"CloudFront-Is-Desktop-Viewer\": \"true\",\"CloudFront-Is-Mobile-Viewer\": \"false\",
\"CloudFront-Is-SmartTV-Viewer\": \"false\",\"CloudFront-Is-Tablet-Viewer\": \"false\",\"CloudFront-Viewer-Country\": \"US\",\"content-type\": \"application/json\",\"Host\": \"jef18742ajw.
execute-api.us-west-2.amazonaws.com\",\"origin\": \"https://cdn.foo.com\",\"Referer\": \"https://cdn.foo.com/bugger/index.html\",\"User-Agent\":\"Mozilla/5.0 (Linux; U; Android 2.3.4;
ko-kr; HTC_X515E Build/GRJ22) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1\",\"Via\": \"2.0 o770SHO0yVumTIPw5M76GC1N2y0TKX0lp.cloudfront.net (CloudFront)\",
\"X-Amz-Cf-Id\": \"lJsX2VQF96vKeckdvtiYbr86NJPI3qa6NJkbSP9g7N26FpZM0mUd0kCO\",\"X-Amzn-Trace-Id\": \"Root=0UmGwfkeveeli7Eo62x6IgiCIya9ai6KMj\",\"X-Forwarded-For\": \"12.130.60.5\",
\"X-Forwarded-Port\": \"443\",\"X-Forwarded-Proto\": \"https\"},\"multiValueHeaders\": {\"Accept\": [\"application/json, text/javascript, */*; q=0.01\"],\"Accept-Encoding\": [\"gzip,
deflate, br\"],\"Accept-Language\": [\"en-US,en;q=0.9\"],\"CloudFront-Forwarded-Proto\": [\"https\"],\"CloudFront-Is-Desktop-Viewer\": [\"true\"],\"CloudFront-Is-Mobile-Viewer\":
[\"false\"],\"CloudFront-Is-SmartTV-Viewer\": [\"false\"],\"CloudFront-Is-Tablet-Viewer\": [\"false\"],\"CloudFront-Viewer-Country\": [\"US\"],\"content-type\": [\"application/json\"],
\"Host\": [\"jef18742ajw.execute-api.us-west-2.amazonaws.com\"],\"origin\": [\"https://cdn.foo.com\"],\"Referer\": [\"https://cdn.foo.com/bugger/index.html\"],\"User-Agent\": [\"Mozilla/5.
0 (Linux; U; Android 2.3.4; ko-kr; HTC_X515E Build/GRJ22) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1\"],\"Via\": [\"2.0 SApGYuP694Zi7bfaDMM8mTyltSJCWY4sH.
cloudfront.net (CloudFront)\"],\"X-Amz-Cf-Id\": [\"4FfvFRSl2FGhY0sel07vsqdg79Xhd3DEjG9nPThyEsTAbPDckiAnsdPO\"],\"X-Amzn-Trace-Id\": [\"Root=tkMgGOpT3d0DZ53KA3ulTUJsAZuIGFjdFo\"],
\"X-Forwarded-For\": [\"12.130.60.5\"],\"X-Forwarded-Port\": [\"443\"],\"X-Forwarded-Proto\": [\"https\"]},\"queryStringParameters\": null,\"multiValueQueryStringParameters\": null,
\"pathParameters\": null,\"stageVariables\": null,\"requestContext\": {\"resourceId\": \"vE6wyI\",\"resourcePath\": \"/done\",\"httpMethod\": \"POST\",\"extendedRequestId\":
\"we9PBiOnJpeqbPS5\",\"requestTime\": \"23/Apr/2020:13:37:40 +0000\",\"path\": \"/prod/done\",\"accountId\": \"718298359019\",\"protocol\": \"HTTP/1.1\",\"stage\": \"prod\",
\"domainPrefix\": \"jef18742ajw\",\"requestTimeEpoch\": 1587649060},\"requestId\": \"a8f641b5-007a-44ba-b013-391ffee8cbdf\",\"identity\": {\"cognitoIdentityPoolId\": null,\"accountId\":
null,\"cognitoIdentityId\": null,\"caller\": null,\"sourceIp\": \"12.130.60.5\",\"accessKey\": null,\"cognitoAuthenticationType\": null,\"cognitoAuthenticationProvider\": null,\"userArn\":
null,\"userAgent\": \"Mozilla/5.0 (Linux; U; Android 2.3.4; ko-kr; HTC_X515E Build/GRJ22) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1\",\"user\": null},
\"domainName\": \"jef18742ajw.execute-api.us-west-2.amazonaws.com\",\"apiId\": \"jef18742ajw\"},\"body\": \"{\\\"done\\\":\\\"true\\\"}\",\"isBase64Encoded\": false","_time":1587649060}
{"_raw":"{\"resource\": \"/done\",\"path\": \"/done\",\"httpMethod\": \"POST\",\"headers\":{\"Accept\":\"application/json, text/javascript, */*; q=0.01\",\"Accept-Encoding\":\"gzip,
deflate, br\",\"Accept-Language\": \"en-US,en;q=0.9\",\"CloudFront-Forwarded-Proto\": \"https\",\"CloudFront-Is-Desktop-Viewer\": \"true\",\"CloudFront-Is-Mobile-Viewer\": \"false\",
\"CloudFront-Is-SmartTV-Viewer\": \"false\",\"CloudFront-Is-Tablet-Viewer\": \"false\",\"CloudFront-Viewer-Country\": \"US\",\"content-type\": \"application/json\",\"Host\": \"jef18742ajw.
execute-api.us-west-2.amazonaws.com\",\"origin\": \"https://cdn.foo.com\",\"Referer\": \"https://cdn.foo.com/bugger/index.html\",\"User-Agent\":\"Mozilla/5.0 (Linux; U; Android 2.3.4;
en-us; DROID BIONIC 4G Build/5.5.1_84_DBN-55) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1\",\"Via\": \"2.0 Gxfnv1P2BYONOirsy3mbYJ3mQ1Tp1VE6z.cloudfront.net
(CloudFront)\",\"X-Amz-Cf-Id\": \"0vQgh4R81pf2iU4Ce8Sc4lhuDas5jEd9xKxG6GLhFFtV5JO041XhqTuX\",\"X-Amzn-Trace-Id\": \"Root=8Ih96VyXwjBKfqJhdQXgmF3JhtVj1fAPtS\",\"X-Forwarded-For\": \"27.101.
11.11\",\"X-Forwarded-Port\": \"443\",\"X-Forwarded-Proto\": \"https\"},\"multiValueHeaders\": {\"Accept\": [\"application/json, text/javascript, */*; q=0.01\"],\"Accept-Encoding\":
[\"gzip, deflate, br\"],\"Accept-Language\": [\"en-US,en;q=0.9\"],\"CloudFront-Forwarded-Proto\": [\"https\"],\"CloudFront-Is-Desktop-Viewer\": [\"true\"],\"CloudFront-Is-Mobile-Viewer\":
[\"false\"],\"CloudFront-Is-SmartTV-Viewer\": [\"false\"],\"CloudFront-Is-Tablet-Viewer\": [\"false\"],\"CloudFront-Viewer-Country\": [\"US\"],\"content-type\": [\"application/json\"],
\"Host\": [\"jef18742ajw.execute-api.us-west-2.amazonaws.com\"],\"origin\": [\"https://cdn.foo.com\"],\"Referer\": [\"https://cdn.foo.com/bugger/index.html\"],\"User-Agent\": [\"Mozilla/5.
0 (Linux; U; Android 2.3.4; en-us; DROID BIONIC 4G Build/5.5.1_84_DBN-55) AppleWebKit/533.1 (KHTML, like Gecko) Version/4.0 Mobile Safari/533.1\"],\"Via\": [\"2.0
Jnnm3wlOvVeAmunA0KOx4SXwiUXulcxFO.cloudfront.net (CloudFront)\"],\"X-Amz-Cf-Id\": [\"cIBTbSBqCviZzvjYupDHxhEPEOa2LWsi1Rb4Wr3n8VHIIXsA2wngpWXc\"],\"X-Amzn-Trace-Id\":
[\"Root=cOg5BoFyNfidslrtQoantehkhe4ks5UwNX\"],\"X-Forwarded-For\": [\"27.101.11.11\"],\"X-Forwarded-Port\": [\"443\"],\"X-Forwarded-Proto\": [\"https\"]},\"queryStringParameters\": null,
\"multiValueQueryStringParameters\": null,\"pathParameters\": null,\"stageVariables\": null,\"requestContext\": {\"resourceId\": \"U1EEYH\",\"resourcePath\": \"/done\",\"httpMethod\":
\"POST\",\"extendedRequestId\": \"lNJoleEkVB7wt8xG\",\"requestTime\": \"23/Apr/2020:13:37:40 +0000\",\"path\": \"/prod/done\",\"accountId\": \"718298359019\",\"protocol\": \"HTTP/1.1\",
\"stage\": \"prod\",\"domainPrefix\": \"jef18742ajw\",\"requestTimeEpoch\": 1587649061},\"requestId\": \"62f87d2a-7bab-471c-8fe9-5c961407f3c2\",\"identity\": {\"cognitoIdentityPoolId\":
null,\"accountId\": null,\"cognitoIdentityId\": null,\"caller\": null,\"sourceIp\": \"27.101.11.11\",\"accessKey\": null,\"cognitoAuthenticationType\": null,
\"cognitoAuthenticationProvider\": null,\"userArn\": null,\"userAgent\": \"Mozilla/5.0 (Linux; U; Android 2.3.4; en-us; DROID BIONIC 4G Build/5.5.1_84_DBN-55) AppleWebKit/533.1 (KHTML,
```

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:



Sample data file: capture_20231026-133818.log
Pipeline: palo_alto_traffic

Show Dropped Events: on
Show Internal Fields: off
Enable Diff: on
Render Whitespace: off
Timeout (sec): 10

```
1   α _raw: 2023-10-26 11:37:44,023,Event [Event=UpdateBillingProvQuo
            erties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.9B107
2023-10-26  hangeESN, quotePriority... Show more
13:37:44.023
+02:00  # _time: 1698320264.023
        α host: localhost
        α source: cribl
        α sourcetype: cribl
2   α _raw: 2023-10-26 11:37:44,023,Event [Event=UpdateBillingProvQuo
            erties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.246A5
2023-10-26  ewActivation, quotePrio... Show more
13:37:44.023
+02:00  # _time: 1698320264.023
        α host: localhost
        α source: cribl
        α sourcetype: cribl
3   α _raw: 2023-10-26 11:37:44,023,Event [Event=UpdateBillingProvQuote, timestamp=1581426263, prop
```

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

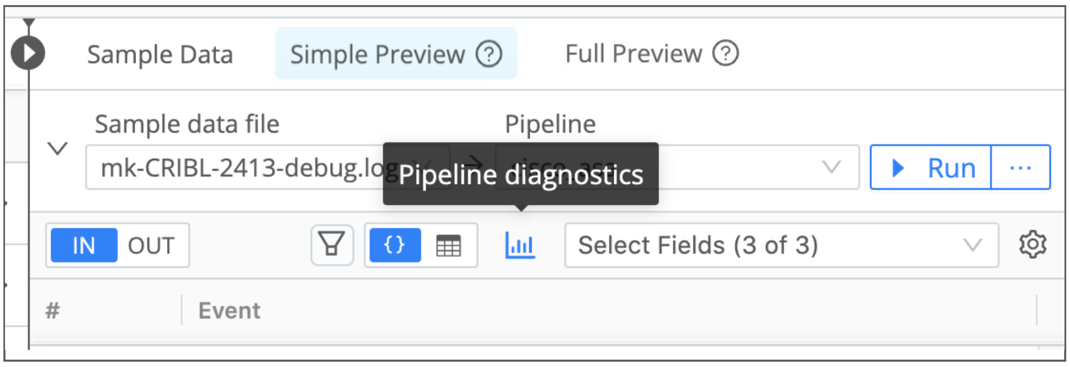
This toggles between displaying carriage returns, newlines, tabs, and spaces as white space, versus as(respectively) the symbols ␍ , ␊ , →, and · .

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.

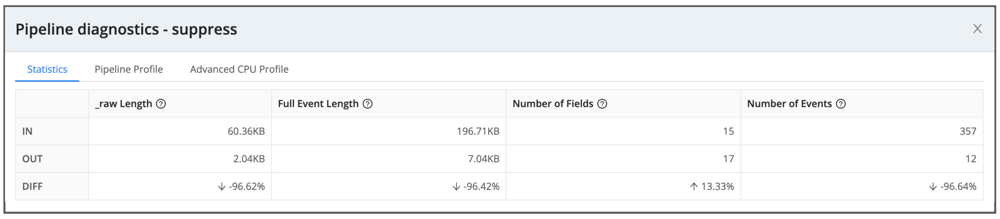
Pipeline Profiling

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

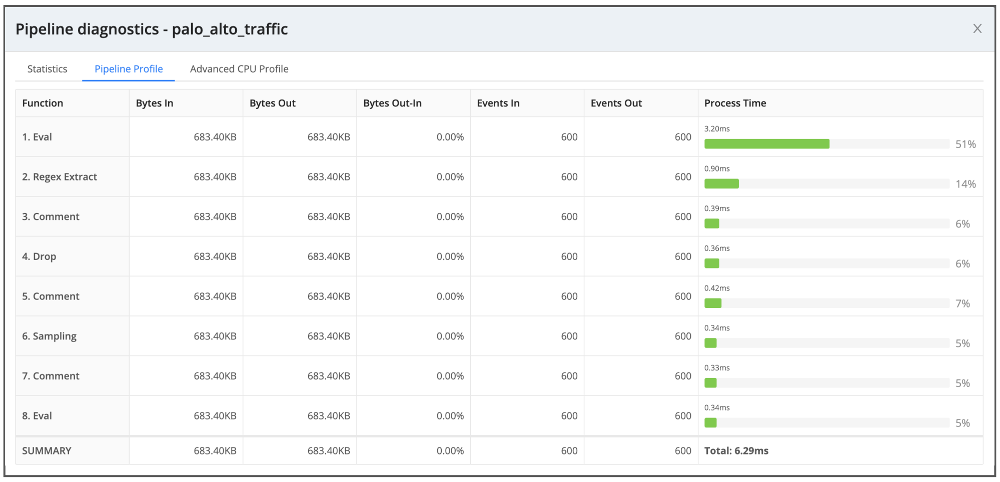
The resulting modal's **Statistics** tab displays basic stats about the Pipeline's processing impact on field and event length and counts.



	_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. Every Function has a minimum processing time, including Functions that make no changes to event data, such as the Comment Function.



Pipeline diagnostics - palo_alto_traffic

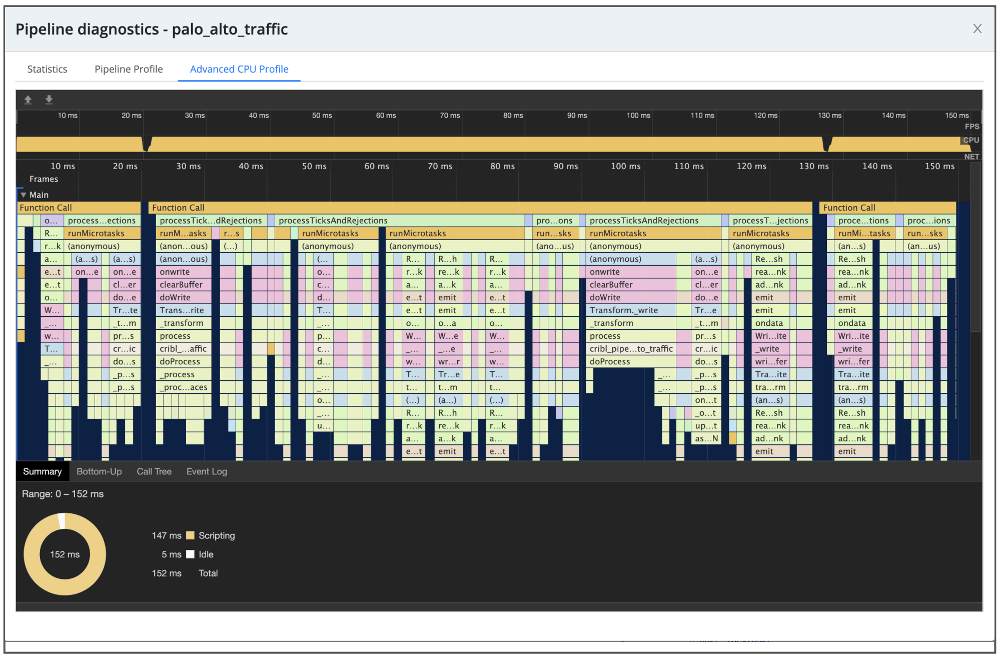
Statistics | Pipeline Profile | Advanced CPU Profile

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

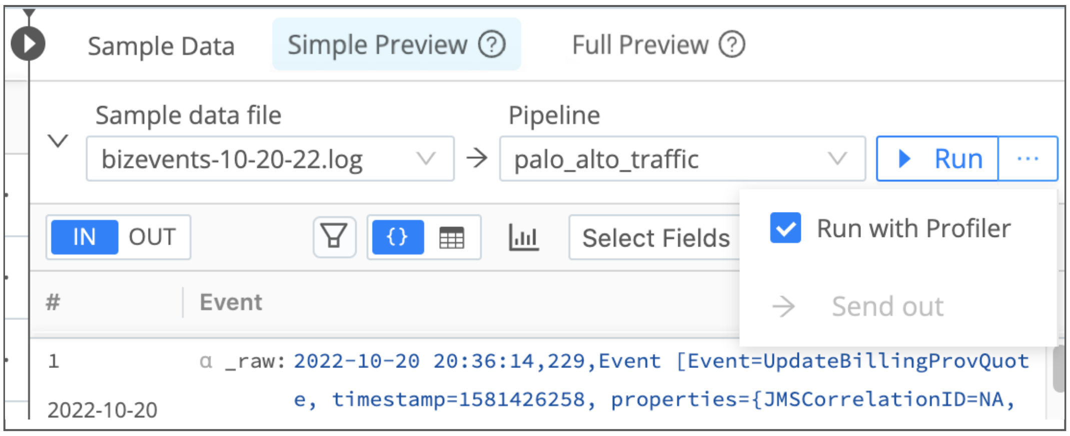
When you profile a Chain Function, this tab condenses all of the chained Pipeline's processing into a single **Chain** row. To see individual statistics on the aggregated Functions, profile the chained Pipeline separately.

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.

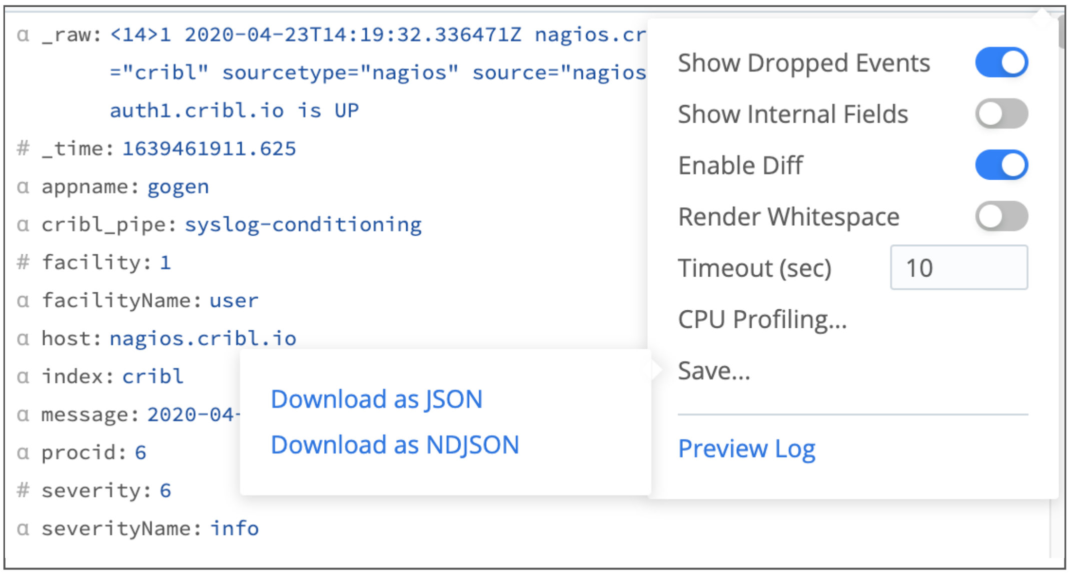


```
1           α _raw: 2022-10-20 20:36:14,229,Event [Event=UpdateBillingProvQuot
2022-10-20          e, timestamp=1581426258, properties={JMSCorrelationID=NA,
```

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.



```
α _raw: <14>1 2020-04-23T14:19:32.336471Z nagios.cr
       ="cribl" sourcetype="nagios" source="nagios
       auth1.cribl.io is UP
# _time: 1639461911.625
α appname: gogen
α cribl_pipe: syslog-conditioning
# facility: 1
α facilityName: user
α host: nagios.cribl.io
α index: cribl
α message: 2020-04-
α procid: 6
# severity: 6
α severityName: info
```

Saving sample data as JSON

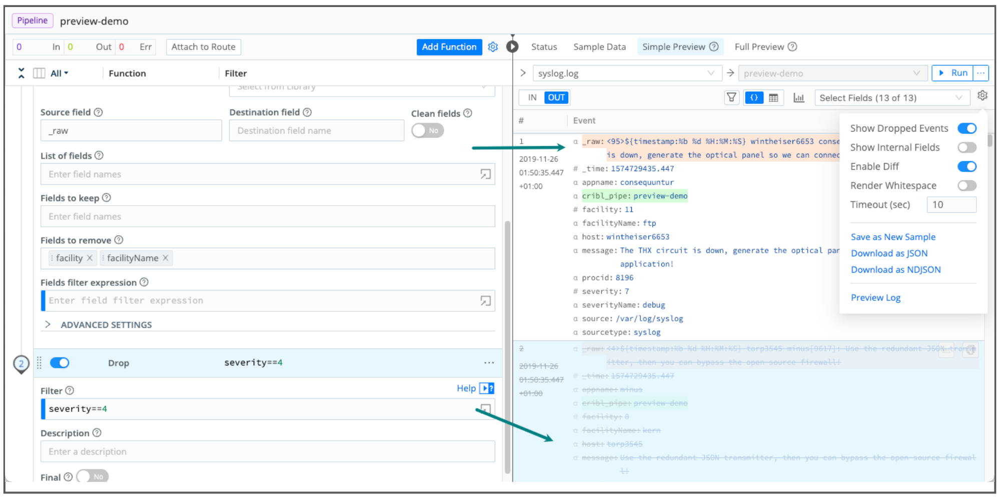
Preview Log

The final option on the Advanced Settings menu opens a modal where you can preview Cribl Stream'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** toggle.
- **Added fields**: When Cribl Stream'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.



```
#   Event
1   _raw: <95>${timestamp:%b %d %H:%M:%S} wintheiser6653 conse
        is down, generate the optical panel so we can connec
2019-11-26
01:50:35.447  _time: 1574729435.447
+01:00        appname: consequuntur
              cribl_pipe: preview-demo
              facility: 11
              facilityName: ftp
              host: wintheiser6653
              message: The THX circuit is down, generate the optical par
                  application!
              procid: 8196
              severity: 7
              severityName: debug
              source: /var/log/syslog
              sourcetype: syslog
```

~~2~~ ~~_raw: <4>${timestamp:%b %d %H:%M:%S} torp3545 minus[9617]: Use the redundant JSON tran itter, then you can bypass the open-source firewall!~~
~~2019-11-26 01:50:35.447 +01:00~~
~~_time: 1574729435.447~~
~~appname: minus~~
~~cribl_pipe: preview-demo~~
~~facility: 0~~
~~facilityName: kern~~
~~host: torp3545~~
~~message: Use the redundant JSON transmitter, then you can bypass the open-source firewal l!~~

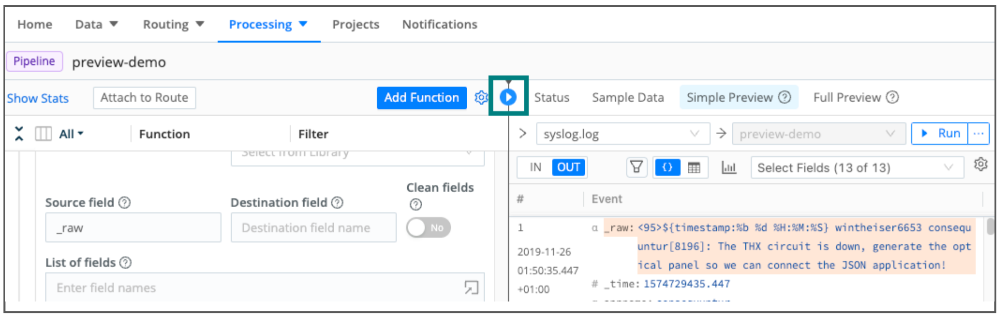
Dropped and added fields in a Pipeline's output

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 Stream adds to events, as well as Source-specific fields that Cribl Stream forwards from upstream senders.

Managing the Preview Pane

On the pane divider between Route or Pipeline and the preview, 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.)



```
1            α _raw: <95>${timestamp:%b %d %H:%M:%S} wintheiser6653 consequ
2019-11-26       untur[8196]: The THX circuit is down, generate the opt
01:50:35.447     ical panel so we can connect the JSON application!
+01:00       # _time: 1574729435.447
```

Collapse / Expand toggle

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.

13.8. Data Onboarding

Onboarding data into Cribl Stream 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 Stream 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 Stream.

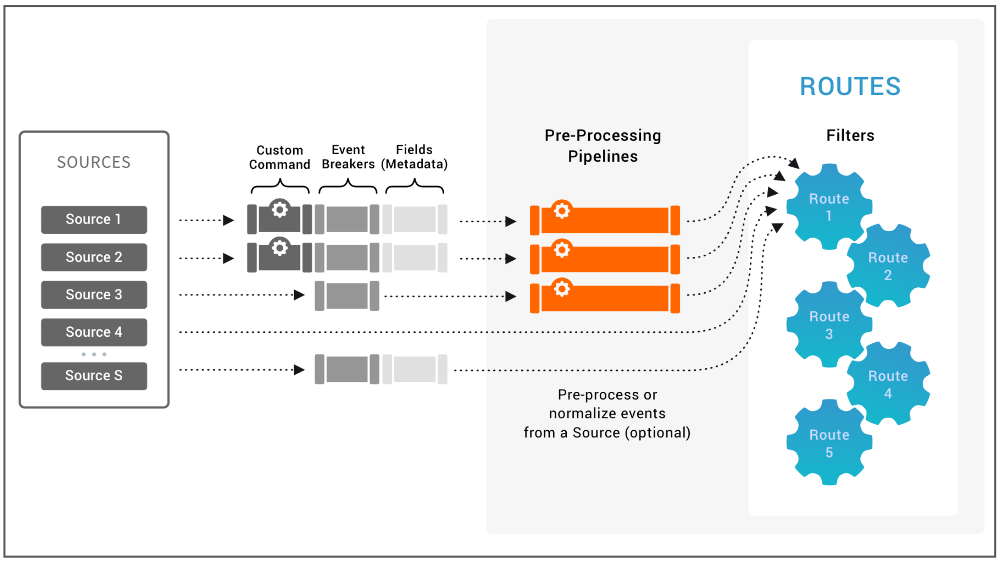
You can also obtain sample data in a live data capture from a Source. One way to do this **before** going to production is to configure your Source with a **devnull** Pipeline (which just drops all events) as a pre-processing Pipeline. Then, let data flow in for just long enough to capture a sufficient sample.

Very Large Integer Values

Cribl Stream'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 Stream 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 Stream Routes.



Source-level processing options

Custom Command

Where supported, data streams will be handled by **custom commands**. These are external system commands that can (optionally) be used to pre-process the data. You can specify any command, script, etc., that consumes via `stdin` and outputs via `stdout`.

Verify that such commands are doing what's expected, as they are the very **first** in a series of processing steps.

Event Breakers

Next, data streams are handled by Event Breakers, which:

- Convert data streams into discrete events.
- Extract and assign timestamps to each event.

If the resulting events do not look correct, feel free to use **non-default** breaking rules and timestamp recognition patterns. Downstream, you can use the Auto Timestamp Function to modify `_time` as needed, if timestamps were not recognized properly. Examples of such errors are:

- Timestamps too far out in the future or past
- Wrong timezone.
- Incorrect timestamp is selected from multiple timestamps present in the event.

Fields

Next, events can be enriched with Fields . This is where you'd add static or dynamic fields to all events delivered by a particular Source.

Pre-Processing Pipeline

Next, you can optionally configure a pre-processing Pipeline on a particular Source. This is extremely useful in these cases:

- Drop non-useful events as early as possible (so as to save on CPU processing).
- Normalize events from this Source to conform a certain shape or structure.
- Fix/touch up events accordingly. E.g., if event breakers assigned the wrong timestamp, this is the best place to use the Auto Timestamp Function to adjust `_time`.

We Can't Say This Enough

Verify, verify, verify, data integrity before turning on the Source.

After Turning On the Source

Use data Destinations to verify that certain metrics of interest are accurate. This will depend significantly on the capabilities of each Destination, but here's a basic checklist of things to ensure:

- Timestamps are correct.
- All necessary fields are assigned to events.
- All expected events show up correctly. (E.g., if a Drop or Suppress Function was configured, ensure that it's not dropping unintended events.)
- Throughput – both in bytes and in events per second (EPS) – is what's expected, or is within a certain tolerance.

Iterate

Iterate on the steps above as necessary. E.g., adjust fields values and timestamps as needed.

Remember that there is almost always a workaround. Any arbitrary event transformation that you need is likely just a Function or two away.

14. Functions

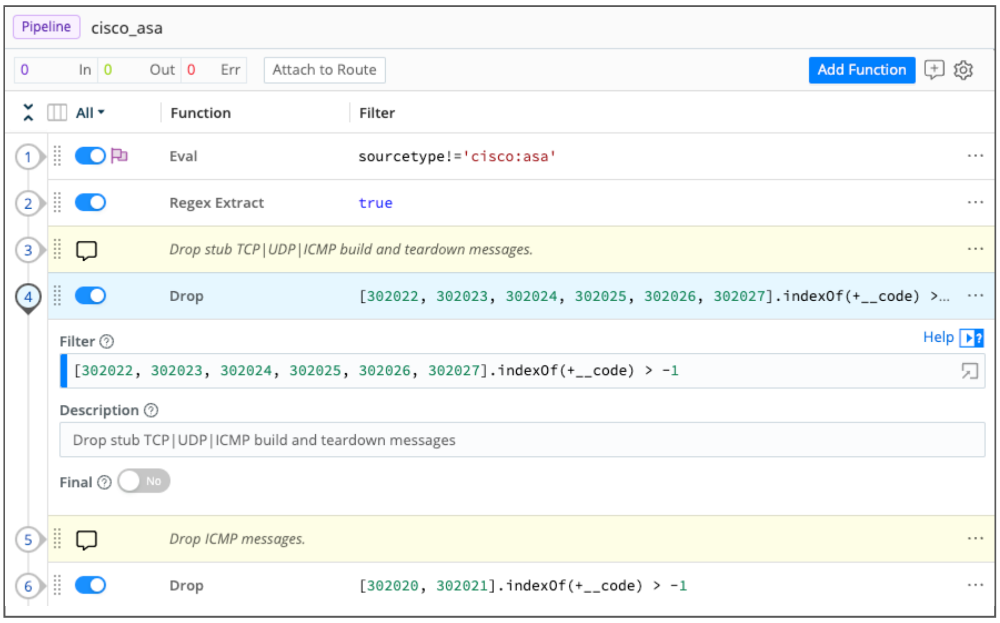
When events enter a Pipeline, they're processed by a series of Functions. At its core, a Function is code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event.

The term "processing" means a variety of possible options: string replacement, obfuscation, encryption, event-to-metrics conversions, etc. For example, a Pipeline can be composed of several Functions – one that replaces the term `foo` with `bar`, another one that hashes `bar`, and a final one that adds a field (say, `dc=jfk-42`) to any event that matches `source=='us-nyc-application.log'`.

How Do They Work

Functions are atomic pieces of JavaScript code that are invoked on each event that passes through them. To help improve performance, Functions can be configured with filters to further scope their invocation to matching events only.

You can add as many Functions in a Pipeline as necessary, though the more you have, the longer it will take each event to pass through. Also, you can enable and disable Functions within a Pipeline as necessary. This lets you preserve structure as you optimize or debug.



	Function	Filter
1	Eval	`sourcetype!='cisco:asa'`
2	Regex Extract	`true`
3	*Drop stub TCP|UDP|ICMP build and teardown messages.*	
4	Drop	`[302022, 302023, 302024, 302025, 302026, 302027].indexOf(+__code) >...`
5	*Drop ICMP messages.*	
6	Drop	`[302020, 302021].indexOf(+__code) > -1`

Filter: `[302022, 302023, 302024, 302025, 302026, 302027].indexOf(+__code) > -1`

Description: Drop stub TCP|UDP|ICMP build and teardown messages

Functions stack in a Pipeline

You can reposition Functions up or down the Pipeline stack to adjust their execution order. Use a Function's left grab handle to drag and drop it into place.

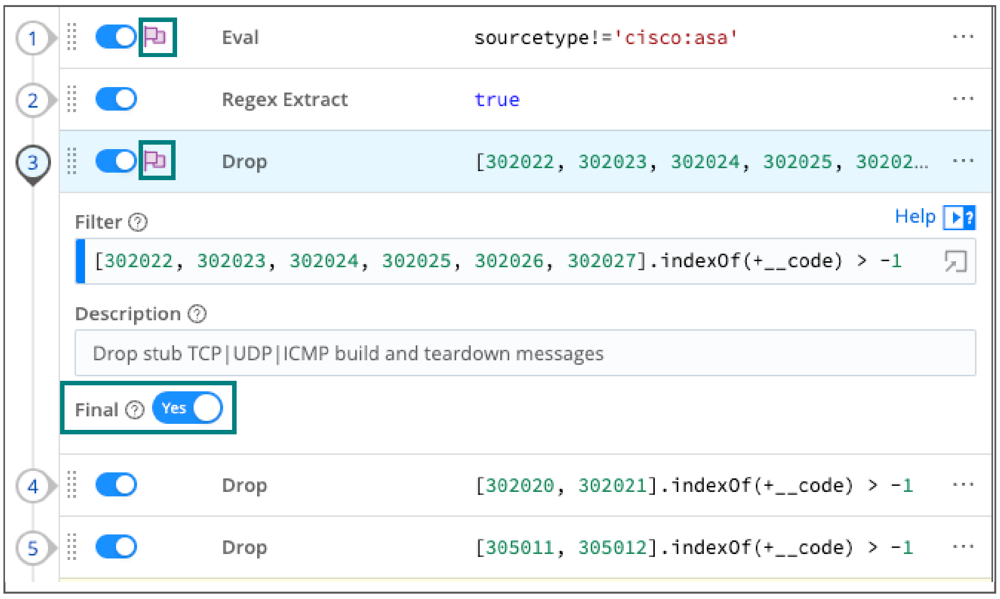
The `Final` Toggle

The `Final` toggle in Function settings controls what happens to the results of a Function.

When `Final` is toggled to `No` (default), events will be processed by this Function, **and** then passed on to the next Function below.

When `Final` is toggled to `Yes`, the Function “consumes” the results, meaning they will not pass down to any other Function below.

A flag in Pipeline view indicates that a Function is `Final`.



	Function	Filter
1	Eval	`sourcetype!='cisco:asa'`
2	Regex Extract	`true`
3	Drop	`[302022, 302023, 302024, 302025, 30202…`
4	Drop	`[302020, 302021].indexOf(+__code) > -1`
5	Drop	`[305011, 305012].indexOf(+__code) > -1`

Filter: `[302022, 302023, 302024, 302025, 302026, 302027].indexOf(+__code) > -1`

Description: Drop stub TCP|UDP|ICMP build and teardown messages

Final: Yes

The Eval and first Drop Functions bearing the Final flag

Functions and Shared-Nothing Architecture

Cribl Stream 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 Stream 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 Stream 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 Stream supports the ECMAScript® 2015 Language Specification.

Very Large Integer Values

Cribl Stream'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 Stream 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)
- Transforms dimensional metrics events into the OTLP (OpenTelemetry Protocol) format: OTLP Metrics
- Resolve hostname from IP address: Reverse DNS (beta)
- Extract numeric values from event fields, converting them to type `number`: Numerify
- Send events out to a command or a local file, via `stdin`, from any point in a Pipeline: Tee
- Convert an XML event's elements into individual events: XML Unroll
- Duplicate events in the same Pipeline, with optional added fields: Clone
- Break events **within**, instead of before they reach, a Pipeline: Event Breaker
- Add a text comment within a Pipeline's UI, to label steps without changing event data: Comment

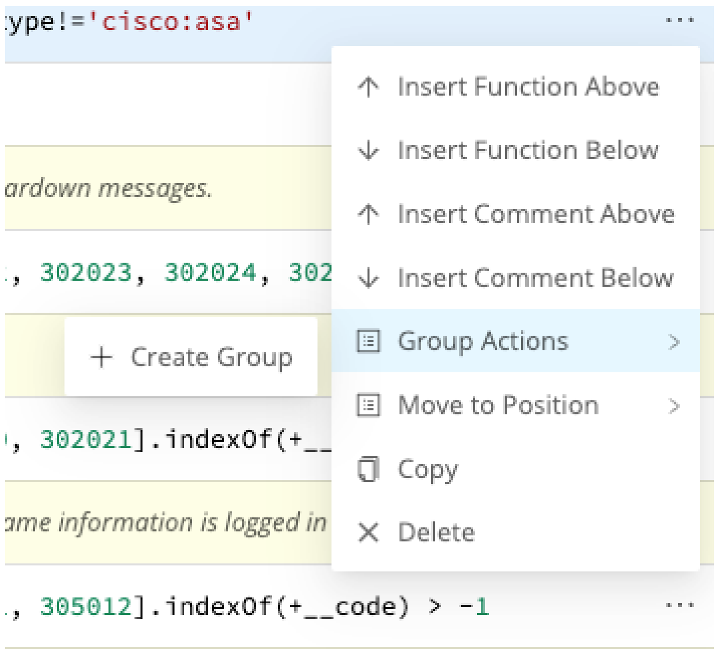
For more usage examples, download Cribl's Stream Cheat Sheet (Quick Reference Card).

Function Groups

A Function group is a collection of consecutive Functions that can be moved up and down a Pipeline's Functions stack together. Groups help you manage long stacks of Functions by streamlining their display. They are a UI visualization only: While Functions are in a group, those Functions maintain their global position order in the Pipeline.

Function groups work much like Route groups.

To build a group from any Function, click the Function's ••• (Options) menu, then select **Group Actions** > **Create Group**.

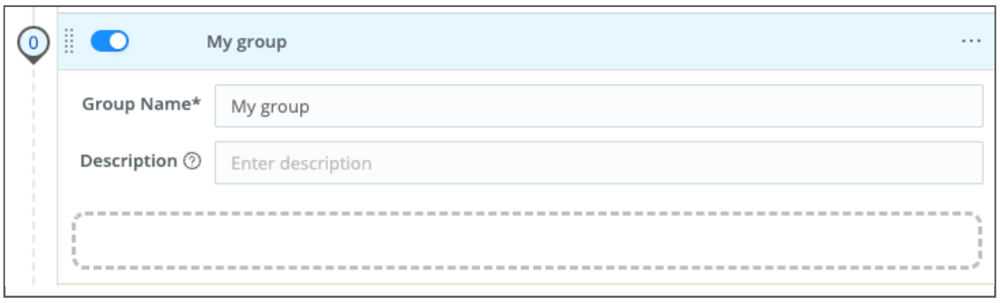


Creating a group

You'll need to enter a **Group Name** before you can save or resave the Pipeline. Optionally, enter a **Description**.

Once you've saved at least one group to a Pipeline, other Functions' ••• (Options) > **Group Actions** submenus will add options to **Move to Group** or **Ungroup/Ungroup All**.

You can also use a Function's left grab handle to drag and drop it into, or out of, a group. A saved group that's empty displays a dashed target into which you can drag and drop Functions.



Drag-and-drop target

14.1. 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 Stream'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
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
```

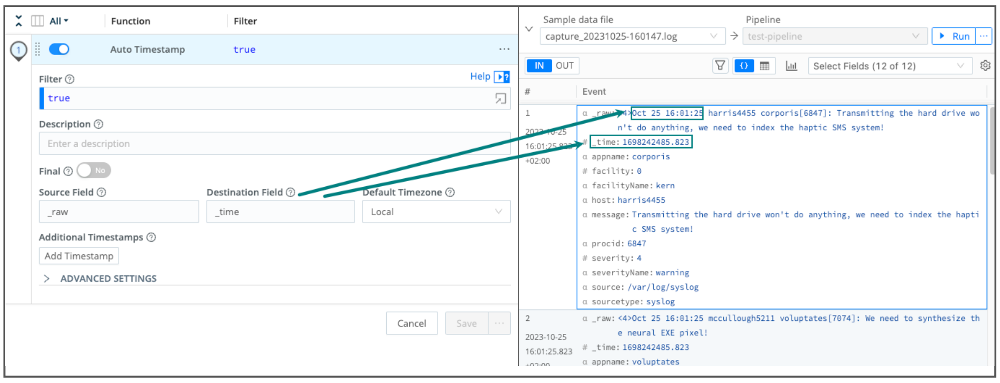
To add this sample (after creating an Auto Timestamp Function with the above **Filter** expression): Go to **Preview** > **Add a Sample** > **Paste a Sample**, and add the data snippet above. Do not make any changes to timestamping or line breaking, and select **Save as Sample File**.

By default, Cribl Stream 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 Stream will grab the first part of the event, and will settle on the first matching value to display for `time`:

- `_time 1698242485.823`
- **GMT**: Oct 25 14:01:25 GMT
- **Your Local Time**: Oct 25 16:01:25 GMT +02:00

Because no explicit timezone has been set (under **Default Timezone**), `_time` will inherit the **Local** timezone, which in this example is `GMT +02:00`.



```
1
2023-10-25
16:01:25.823
+02:00
_raw: <4>Oct 25 16:01:25 harris4455 corporis[6847]: Transmitting the hard drive wo
      n't do anything, we need to index the haptic SMS system!
# _time: 1698242485.823
appname: corporis
# facility: 0
facilityName: kern
host: harris4455
message: Transmitting the hard drive won't do anything, we need to index the hapti
         c SMS system!
procid: 6847
# severity: 4
severityName: warning
source: /var/log/syslog
sourcetype: syslog

2
2023-10-25
16:01:25.823
+02:00
_raw: <4>Oct 25 16:01:25 mccullough5211 voluptates[7074]: We need to synthesize th
      e neural EXE pixel!
# _time: 1698242485.823
appname: voluptates
```

Timezone Dependencies and Details

Cribl Stream 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 Stream will truncate timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.
- For further examples, see Extracting Timestamps from Messy Logs.

14.2. Aggregations

The Aggregations Function performs aggregate statistics on event data.

Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Time window: The time span of the tumbling window for aggregating events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Aggregates: Aggregation function(s) to perform on events.
E.g., `sum(bytes).where(action=='REJECT').as(TotalBytes)`. Expression format: `aggFunction(<FieldExpression>).where(<FilterExpression>).as(<outputField>)`. See more examples below.

When used without `as()`, the aggregate's output will be placed in a field labeled `<fieldName>_<aggFunction>`. If there are conflicts, the last aggregate wins. For example, given two aggregates – `sum(bytes).where(action=='REJECT')` and `sum(bytes)` – the latter one (`bytes_sum`) is the winner.

Group by Fields: Fields to group aggregates by. Supports wildcard expressions.

Evaluate fields: Set of key-value pairs to evaluate and add/set. Fields are added in the context of an aggregated event, before they're sent out. Does not apply to passthrough events.

Time Window Settings

Cumulative aggregations: If enabled, aggregations will be retained for cumulative aggregations when flushing out an aggregation table event. When set to `No` (the default), aggregations will be reset to `0` on flush.

Lag tolerance: The lag tolerance represents the tumbling window tolerance to late events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Idle bucket time limit: The amount of time to wait before flushing a bucket that has not received events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Output Settings

Passthrough mode: Determines whether to pass through the original events along with the aggregation events. Defaults to `No`.

Metrics mode: Determines whether to output aggregates as metrics. Defaults to `No`, causing aggregates to be output as events.

Sufficient stats mode: Determines whether to output *only* statistics sufficient for the supplied aggregations. Defaults to `No`, meaning output richer statistics.

Output prefix: A prefix that is prepended to all of the fields output by this Aggregations Function.

Advanced Settings

Aggregation event limit: The maximum number of events to include in any given aggregation event. Defaults to unlimited. Must be at least `1`.

Aggregation memory limit: The memory usage limit to impose upon aggregations. Defaults to unlimited (i.e., the amount of memory available in the system). Accepts numerals with multiple-byte units, like KB, MB, GB, etc. (such: as `4GB`.)

Flush on stream close: If set to `Yes` (the default), aggregations will flush when an input stream is closed. If set to `No`, the Time Window Settings will control flush behavior; this can be preferable in cases like the following:

- Your input data consists of many small files.
- You are sending data to Prometheus. Enabling **Flush on stream close** can send Prometheus multiple aggregations from the same Worker Process for the same time period. Prometheus cannot tell the multiple aggregations apart, and will ingest only the first one.

Aggregation Functions

- `avg(expr:FieldExpression)`: Returns the average `expr` values.

- `count(expr:FieldExpression)`: If `expr` is omitted, returns the number of events received over the time window. If `expr` is given, returns the number of events seen where `expr` evaluates to a value other than null or undefined. For example, with these three events in the same time window:
  - `{ _time: 20, _val: 5 }`
  - `{ _time: 22 }`
  - `{ _time: 24, _val: null }`

  `count()` would return `3`, while `count(_val)` would return `1`.
- `dc(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of `expr`, within a relative error rate. Lower error rates increase the accuracy of the result, at the cost of using more memory. For example, with these three events in the same time window:
  - `{ _time: 20, _name: 'Alice' }`
  - `{ _time: 22, _name: 'Bob' }`
  - `{ _time: 24, _name: 'Alice' }`

  `dc(_name)` would return `2`.
- `distinct_count(expr: FieldExpression, errorRate: number = 0.01)`: Identical to `dc(expr)`.
- `earliest(expr:FieldExpression)`: Returns the earliest (based on `_time`) observed `expr` value.
- `first(expr:FieldExpression)`: Returns the first observed `expr` value.
- `histogram(expr:FieldExpression, buckets: number[])`. Returns the average of the values of `expr` and generates a field with the same name as the aggregated output field, suffixed with `_data`.
  - `buckets` must contain at least one numeric value.

  The `_data` field in the output contains three pieces of information:
  - `_sum` – the sum of all values of `expr`.
  - `_count` – the number of events seen where `expr` evaluates to a value other than null or undefined.
  - `_buckets` – an object whose keys are the buckets defined in the function, and whose values are the number of values that fall in that histogram bucket. In addition to the buckets defined by the user, the object will include a bucket labeled `Infinity`, into which all values will be placed. A value of `x` falls into a histogram bucket if it is less than or equal to the value of the bucket. A value of `45` would fall into a bucket with value `50`, but not one with value `40`. Counts are cumulative; all values counted in a bucket will also be counted in every bucket larger than it. For example, with these three events in the same time window:
    - `{ _time: 20, age: 20 }`
    - `{ _time: 22, age: 25 }`
    - `{ _time: 24, age: 30 }`

`histogram(age, [10, 25, 40])` would result in the following on the output event:

```
{
  age_histogram: 25
  age_histogram_data: {
    _count: 3,
    _sum: 75
    _buckets: {
      10: 0,
      25: 2,
      40: 3,
      Infinity: 3
    }
  }
}
```

- `last(expr:FieldExpression)`: Returns the last observed `expr` value.
- `latest(expr:FieldExpression)`: Returns the latest (based on `_time`) observed `expr` value.
- `list(expr:FieldExpression[, max:number, excludeNulls: boolean = true])`: Returns a list of the observed values of `expr`.
    - Optional `max` parameter limits the number of values returned. If omitted, the default is `100`. If set to `0`, will return all values.
    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `max(expr:FieldExpression)`: Returns the maximum `expr` value.
- `median(expr:FieldExpression)`: Returns the middle value of the sorted parameter.
- `min(expr:FieldExpression)`: Returns the minimum `expr` value.
- `mode(expr: FieldExpression[, excludeNulls: boolean = true])`: Returns the single most frequently encountered `expr` value.
    - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `per_second(expr:FieldExpression)`: Returns the rate of change of the values of `expr` over the aggregate time window. This is equivalent to `rate(expr, '1s')`. For example, with these three events in the same time window:
    - `{ _time: 20, _val: 5 }`
    - `{ _time: 22, _val: 15 }`
    - `{ _time: 24, _val: 35 }`

`per_second(_val)` would give back `(35 - 5) / (24 - 20) = 7.5`, meaning that `_val` increased by 7.5 every second over the time window.

- `perc(level: number, expr: FieldExpression)`: Returns `<level>` percentile value of the numeric `expr` values.
- `rate(expr:FieldExpression, timeString: string = '1s')`: Returns the rate of change of the values of `expr` over the aggregate time window. Calculated as (latest value - earliest value) / (latest timestamp - earliest timestamp) * number of seconds in `timeString`. For example, with these three events in the same time window:
  - `{ _time: 20, _val: 5 }`
  - `{ _time: 22, _val: 15 }`
  - `{ _time: 24, _val: 35 }`

  `rate(_val, '2s')` would give back `(35 - 5) / (24 - 20) * 2 = 15`, meaning that `_val` increased by 15 every 2 seconds over the time window.
- `stdev(expr:FieldExpression)`: Returns the sample standard deviation of the `expr` values.
- `stdevp(expr:FieldExpression)`: Returns the population standard deviation of the `expr` values.
- `sum(expr:FieldExpression)`: Returns the sum of the `expr` values.
- `summary(expr:FieldExpression)[, quantiles: number[]])`: Returns the average of the `expr` values and generates a field with the same name as the aggregate output field, suffixed with `_data`.
  - Optional: `quantiles` values must be between `0` and `1` inclusive.

  The `_data` field in the output contains three pieces of information:
  - `_sum` – the sum of all `expr` values.
  - `_count` – the number of events seen where `expr` evaluates to a value other than null or undefined.
  - `_quantiles` – an object whose keys are the quantiles defined in the function, and whose values are the quantile value across the `expr` values. For example, a quantile key `0.5` and value `500` would indicate that the 50th percentile (or median) of the values seen was 500.
  - For example, with these three events in the same time window:
    - `{ foo: 100 }`
    - `{ foo: 200 }`
    - `{ foo: 300 }`

    `summary(foo, [0.25, 0.5, 0.75])` would result in the following on the output event:

```
{
  foo_summary: 25
  foo_summary_data: {
    _count: 3,
    _sum: 75
    _quantiles: {
      0.25: 100,
      0.5: 150,
      0.75: 225
    }
  }
}
```

- `sumsq(expr:FieldExpression)`: Returns the sum of squares of the `expr` values.
- `top(expr: FieldExpression, count: number[, excludeNulls: boolean = true])`: Returns the most frequently encountered `expr` values, up to `count` number of results.
  - `count` parameter must be a positive integer.
  - Optional `excludeNulls` boolean excludes null and undefined values from results. If included, defaults to `true`.
- `values(expr:FieldExpression[, max:number, errorRate:number, excludeNulls: boolean = true])`: Returns a list of distinct `expr` values.
  - Optional `max` parameter limits the number of values returned; if omitted, the default is `0`, meaning return all distinct values.
  - Optional `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 `expr` values.
- `variancep(expr:FieldExpression)`: Returns the population variance of the `expr` values.

Safeguarding Data

Upon shutdown, Cribl Stream 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.

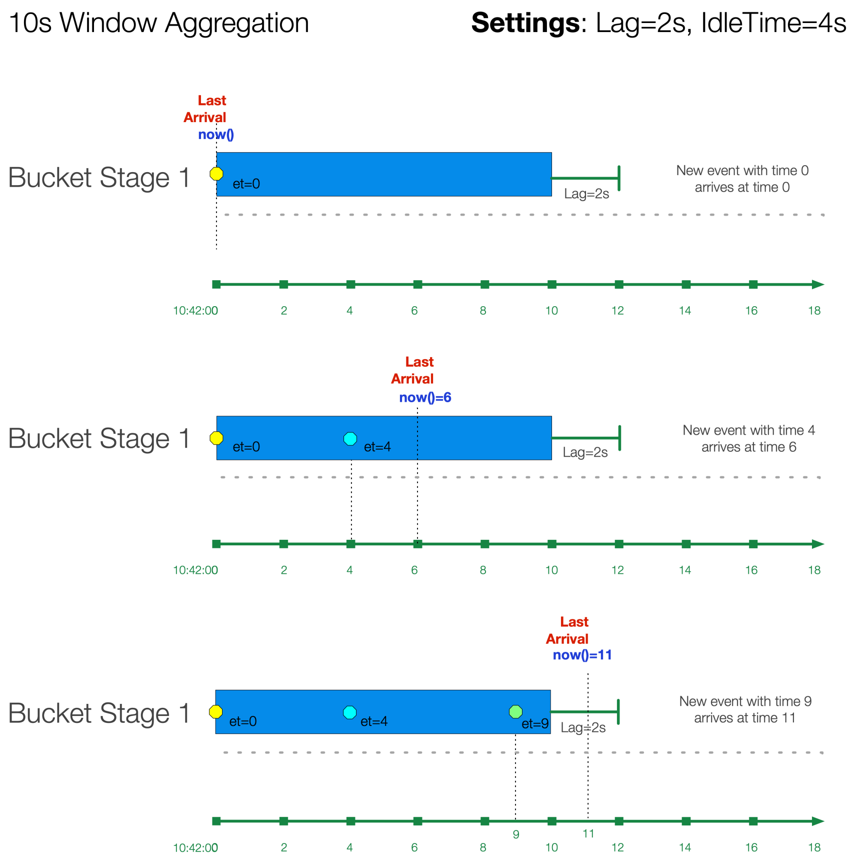
This is especially necessary for high-cardinality data. (Both settings default to unlimited, but we recommend setting defined limits based on testing.)

How Do Time Window Settings Work?

Lag Tolerance

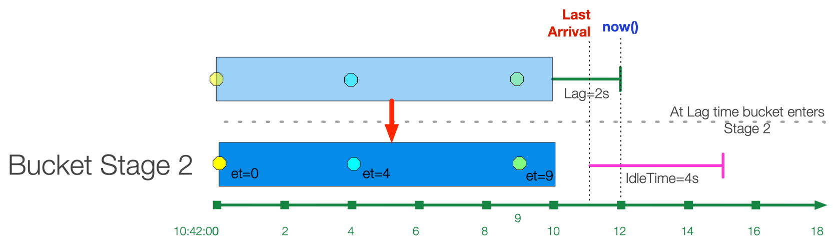
As events are aggregated into windows, there is a good chance that most will arrive later than their event time. For instance, given a `10s` window (`10:42:00 - 10:42:10`), an event with timestamp `10:42:03` might come in 2 seconds later at `10:42:05`.

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.

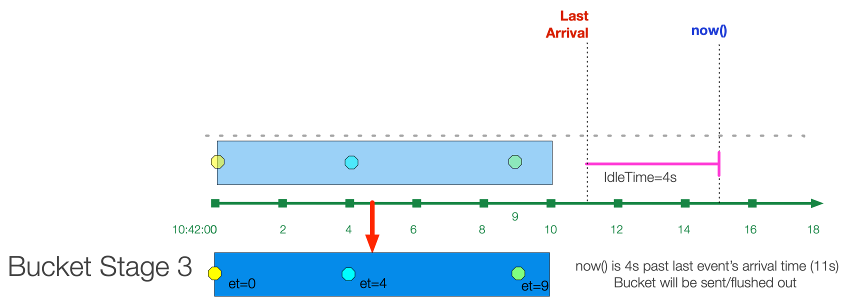
After the lag time expires, the bucket moves to Stage 2.



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.


Last Arrival
now()
IdleTime=4s
10:42:00
2
4
6
8
9
10
12
14
16
18
Bucket Stage 3
et=0
et=4
et=9
now() is 4s past last event's arrival time (11s)
Bucket will be sent/flushed out

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.

14.3. 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

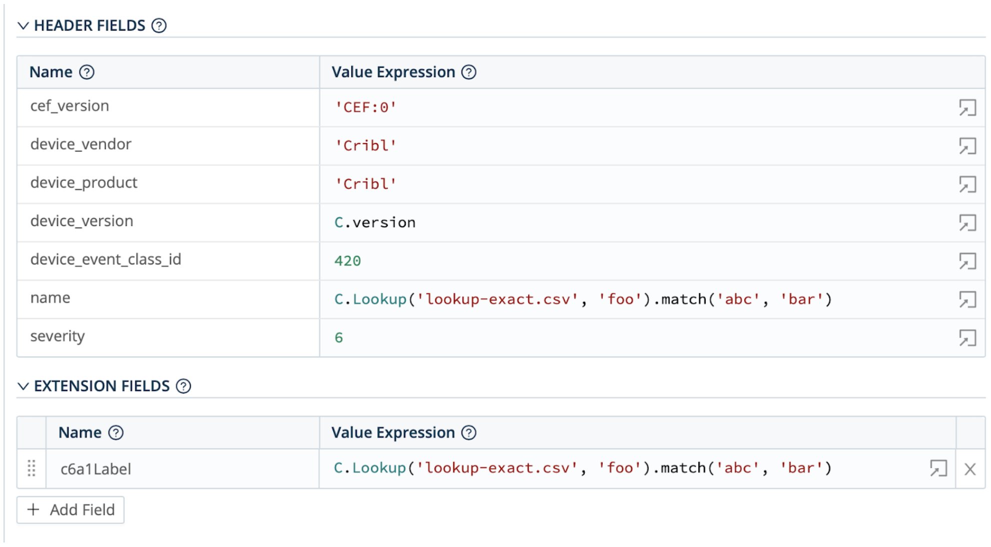
CEF Extension field definitions. Field names and values will be written in `key=value` format. Select each field's Name from the drop-down list. Values can be computed with JS expressions, or can be constants.

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**.



HEADER FIELDS

Name	Value Expression
cef_version	`'CEF:0'`
device_vendor	`'Cribl'`
device_product	`'Cribl'`
device_version	`C.version`
device_event_class_id	`420`
name	`C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')`
severity	`6`

EXTENSION FIELDS

Name	Value Expression
c6a1Label	`C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')`

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
```

14.4. Chain

The Chain Function does one thing: It chains data processing from a Pipeline or Pack to another Pipeline or Pack. This can be useful for sequential processing, or just to separate groups of related Functions into discrete Pipeline or Pack units that make intuitive sense.

Control Flow

The chained Pipeline or Pack will, upon completion, normally return control to the parent Pipeline/Pack containing the Chain Function. However, if the chained Pipeline/Pack contains any Function with the **Final** flag enabled, processing will stop there. In this case, back in the parent Pipeline/Pack, no Function below Chain will execute.

Cycle Detection and Throughput

The Chain Function includes guardrails against circular references. In v.4.3 and later, Cribl Stream detects cycles when you configure your Pipeline. This early detection (compared to earlier versions' runtime detection) means that:

- If you try to save a Pipeline with a cyclical reference, Cribl Stream will throw an error.
- If an existing Pipeline's configuration contains a cycle, Cribl Stream will disable the final Chain Function involved in the cycle.

Despite these safeguards, Cribl recommends that you keep chained configurations simple and understandable by all your users. Also, keep in mind that inserting a Chain Function can impose a slight performance hit, compared to including all processing in the original Pipeline or Pack.

Pipeline Versus Pack Scope

You will see different scope restrictions when using Chain in a Pipeline versus in a Pack:

- In a Pipeline, the **Processor** drop-down displays both Pipelines and Packs as targets to chain to.
- In a Pack, the **Processor** drop-down offers only Pipelines contained within that Pack.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

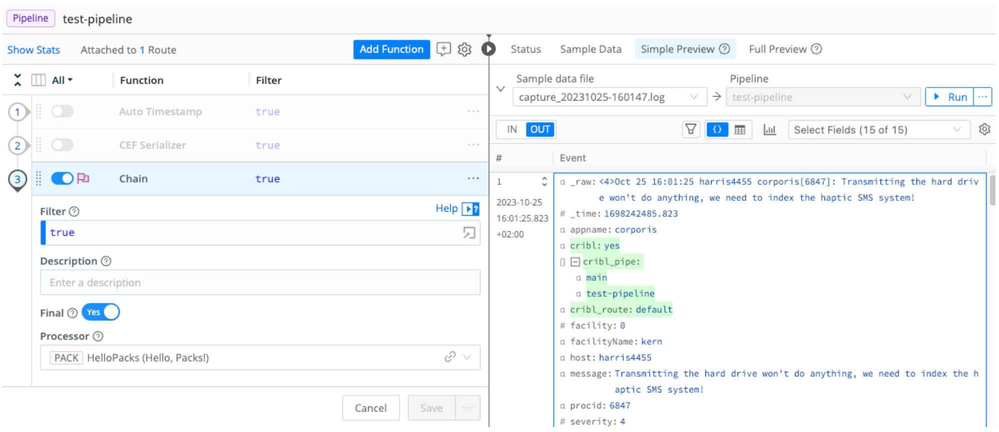
Description: Optionally, add a simple description of this Function’s purpose. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`. (Note that this will not prevent data from flowing to the Function’s defined **Processor**.)

Processor: Use this drop-down to select a configured Pipeline or Pack through which to forward events.

Example

This shows a simple preview of a `pipeline-1` Pipeline, which chains to a `pipeinpipe` Pipeline. Notice that each event’s added `cribl_pipe` field lists all Pipelines/Packs through which the event was chained.



	Function	Filter
1	Auto Timestamp	true
2	CEF Serializer	true
3	Chain	true

Filter: true
Description: Enter a description
Final: Yes
Processor: PACK HelloPacks (Hello, Packs!)

Sample data file: capture_20231025-160147.log
Pipeline: test-pipeline
Select Fields (15 of 15)

```
#   Event
1   _raw: <4>Oct 25 16:01:25 harris4455 corporis[6847]: Transmitting the hard driv
          e won't do anything, we need to index the haptic SMS system!
2023-10-25
16:01:25.823
+02:00
    _time: 1698242485.823
    appname: corporis
    cribl: yes
    cribl_pipe:
      main
      test-pipeline
    cribl_route: default
    facility: 0
    facilityName: kern
    host: harris4455
    message: Transmitting the hard drive won't do anything, we need to index the h
             aptic SMS system!
    procid: 6847
    severity: 4
```

cribl_pipe field shows whole processing path

14.5. 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.

14.6. Code

If you need to operate on data in a way that can't be accomplished with Cribl Stream's out-of-the-box Functions, the Code Function enables you to encapsulate your own JavaScript code. This Function is available in Cribl Stream 3.1+, and imposes some restrictions for security reasons.

Restrictions

Generally speaking, anything forbidden in JavaScript strict mode is forbidden in the context of the Code Function. Specifically, the following are **not allowed**:

- `console`, `eval`, `uneval`, `Function (constructor)`, `Promises`, `setTimeout`, `setInterval`, `global`, `globalThis`, `window`, and `set`.

Code Functions **can** include `for` loops, `while` loops, and JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every`. For further details, see Supported JavaScript Options.

Cribl Stream'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 Stream sends logs from this function to a unique channel in the form `func:code:${pipelineName}:${functionIndex}`. Toggle off to use the generic `func:code` log channel instead.

Notes and Examples

Functions (including the Code Function) always use the special variable `__e` to access the `(context)` event inside JavaScript expressions.

Possibly the simplest Code Function creates a new field and then assigns it a value:

A Basic Code Function

```
__e['foo'] = 'Hello, Goats!'
```

For more ambitious implementations, see Code Function Examples.

JavaScript Support

Cribl Stream 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

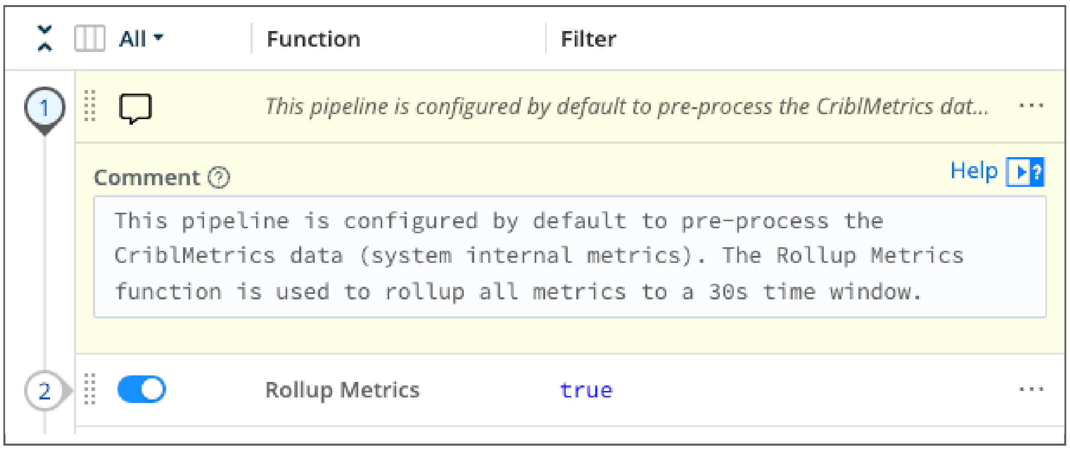
- Indexed collections
- Keyed collections
- Working with Objects

14.7. Comment

The Comment Function adds a text comment in a Pipeline. It makes no changes to event data. The added comment is visible only within the Pipeline UI, where it is useful for labeling Pipeline steps. The Comment Function always has a minimum, non-zero processing time.

Usage

Comment: Add your comment as plain text in this field.

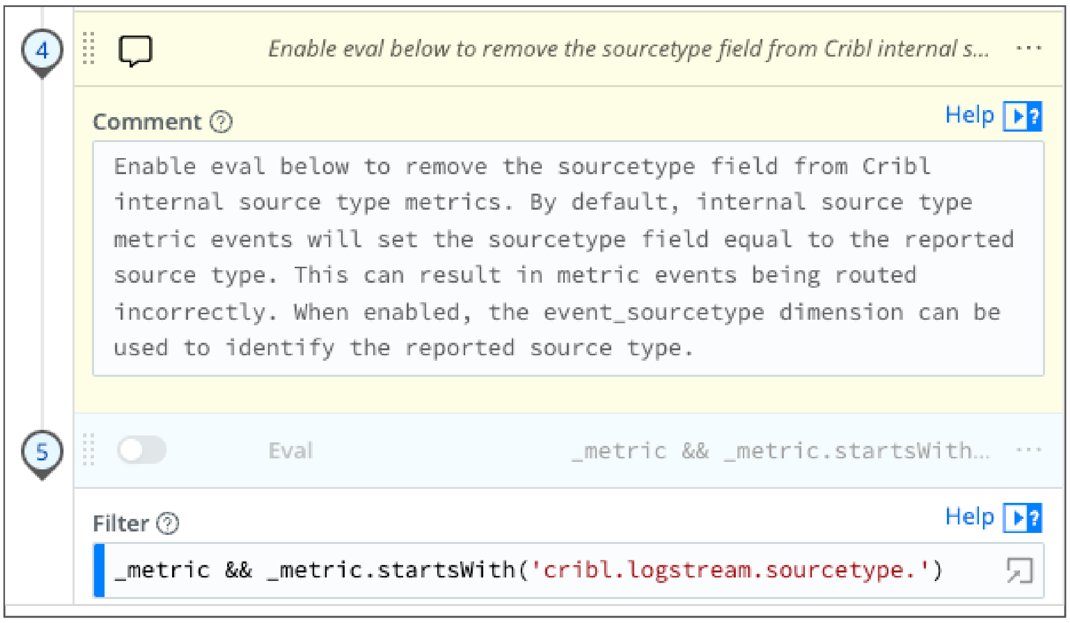


All
Function
Filter
1
This pipeline is configured by default to pre-process the CriblMetrics dat...
Comment
Help
2
Rollup Metrics
true

```
This pipeline is configured by default to pre-process the
CriblMetrics data (system internal metrics). The Rollup Metrics
function is used to rollup all metrics to a 30s time window.
```

Examples

This comment labels the Pipeline's next function:



4
Enable eval below to remove the sourcetype field from Cribl internal s...
Comment
Help
5
Eval
_metric && _metric.startsWith...
Filter
Help

```
Enable eval below to remove the sourcetype field from Cribl
internal source type metrics. By default, internal source type
metric events will set the sourcetype field equal to the reported
source type. This can result in metric events being routed
incorrectly. When enabled, the event_sourcetype dimension can be
used to identify the reported source type.
```

```
_metric && _metric.startsWith('cribl.logstream.sourcetype.')
```

14.8. 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 Stream'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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream. 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 Stream will use the system's DNS server.

Cache time to live (minutes): Determines the interval on which the DNS cache will expire, and its contents will be refetched. Defaults to `30` minutes. Use `0` to disable cache expiration/refresh behavior.

Maximum cache size: Maximum number of DNS resolutions to cache locally. Before changing the default `5000`, consider the implications for your system. Higher values will increase memory usage. Highest allowed value is `100000`.

Monitoring DNS Cache State

When the DNS Lookup Function succeeds in looking up a value it has stored in the DNS cache, that is called a **hit**. Each hit enables the Function to skip contacting a DNS server outside of Cribl Stream to perform the DNS lookup. The resulting performance improvement is the main benefit that the DNS cache provides.

When the DNS Lookup Function looks for a value in the DNS cache, but cannot find it, that is called a **miss**. In this case, the Function must contact a DNS server to perform the lookup.

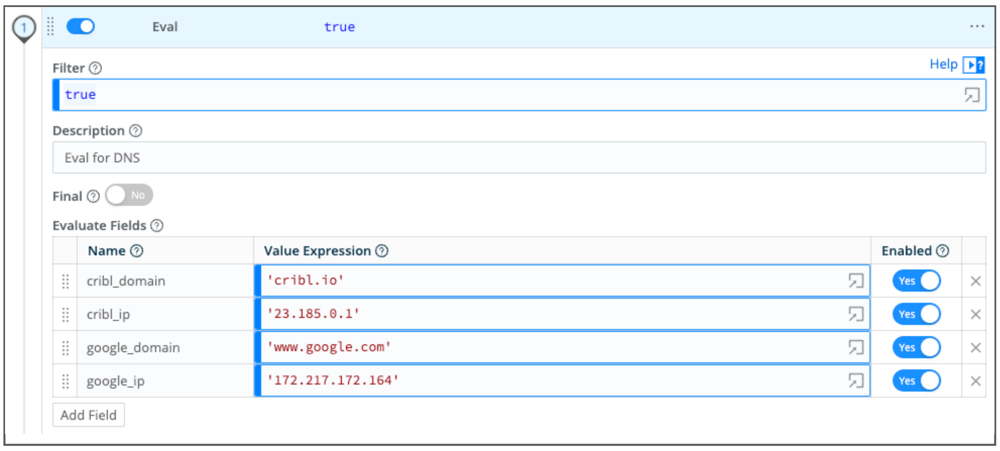
Cribl Stream logs the DNS cache's state in the `func:dns_lookup` channel. The table below explains some typical messages.

Log Message	Meaning
`hits`	Number of successful lookups from cache.
`misses`	Number of unsuccessful lookups from cache.
`keys`	Number of entries in cache.
`ksize`	Approximate amount of memory occupied by keys, in bytes.
`vsize`	Approximate amount of memory occupied by values, in bytes.

See Monitoring for a general explanation of logging in Cribl Stream.

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.


Eval true

Filter: `true`

Description: Eval for DNS

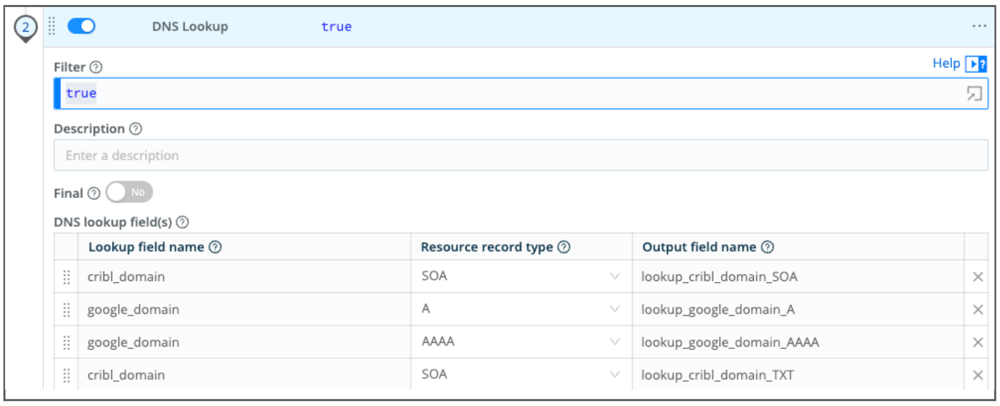
Final: No

Evaluate Fields

Name	Value Expression	Enabled
cribl_domain	`'cribl.io'`	Yes
cribl_ip	`'23.185.0.1'`	Yes
google_domain	`'www.google.com'`	Yes
google_ip	`'172.217.172.164'`	Yes

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 `true`

Filter: `true`

Description: Enter a description

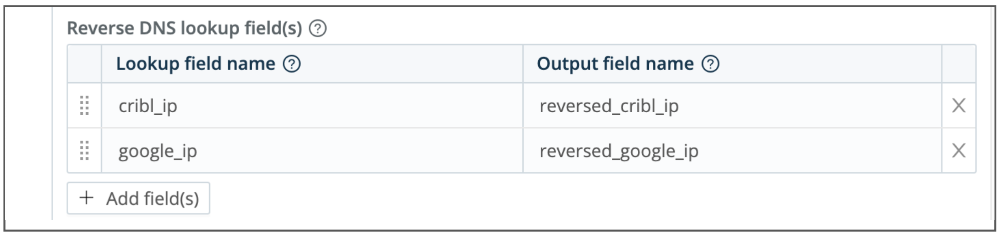
Final: No

DNS lookup field(s)

Lookup field name	Resource record type	Output field name
cribl_domain	SOA	lookup_cribl_domain_SOA
google_domain	A	lookup_google_domain_A
google_domain	AAAA	lookup_google_domain_AAAA
cribl_domain	SOA	lookup_cribl_domain_TXT

DNS Lookup: multiple record types

Finally, the same Function's **Reverse DNS lookup** section retrieves domain names for the two IP addresses.



Reverse DNS lookup field(s)

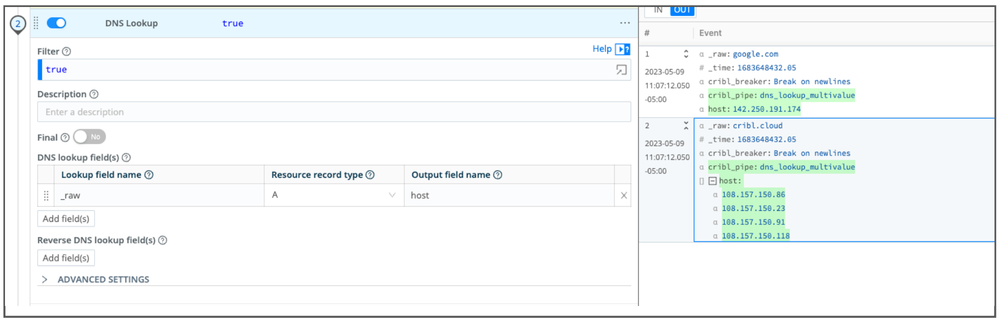
Lookup field name	Output field name
cribl_ip	reversed_cribl_ip
google_ip	reversed_google_ip

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 `true`

Filter: `true`

Description: Enter a description

Final: No

DNS lookup field(s)

Lookup field name	Resource record type	Output field name
_raw	A	host

Reverse DNS lookup field(s)

```
1   2023-05-09 11:07:12.050 -05:00
    _raw: google.com
    _time: 1683648432.05
    cribl_breaker: Break on newlines
    cribl_pipe: dns_lookup_multivalue
    host: 142.250.191.174
2   2023-05-09 11:07:12.050 -05:00
    _raw: cribl.cloud
    _time: 1683648432.05
    cribl_breaker: Break on newlines
    cribl_pipe: dns_lookup_multivalue
    host:
      108.157.150.86
      108.157.150.23
      108.157.150.91
      108.157.150.118
```

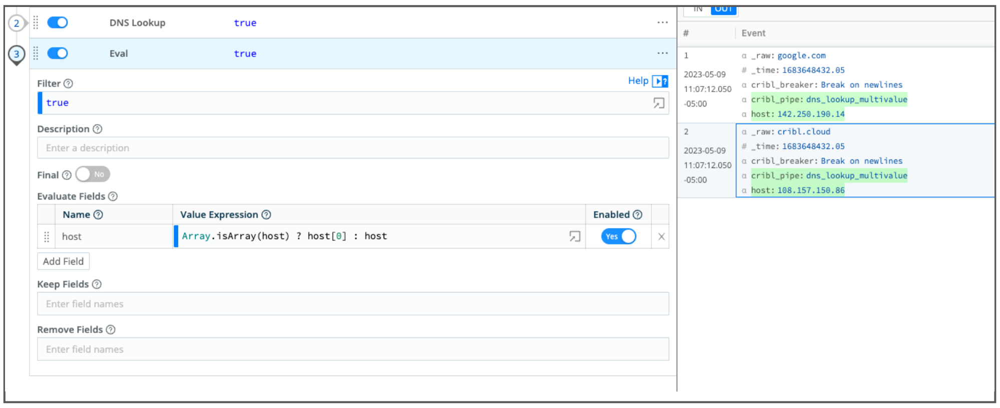
DNS lookup with multi-value result

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.



Filter: `true`

Name	Value Expression	Enabled
host	`Array.isArray(host) ? host[0] : host`	Yes

#	Event
1 2023-05-09 11:07:12.050 -05:00	_raw: google.com _time: 1683648432.05 cribl_breaker: Break on newlines cribl_pipe: dns_lookup_multivalue host: 142.250.190.14
2 2023-05-09 11:07:12.050 -05:00	_raw: cribl.cloud _time: 1683648432.05 cribl_breaker: Break on newlines cribl_pipe: dns_lookup_multivalue host: 108.157.150.86

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.

14.9. 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'`

14.10. 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

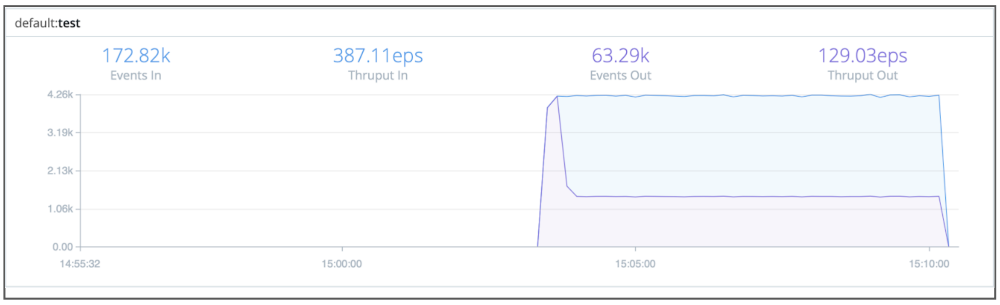
Here's an example that illustrates the effectiveness of using the Square Root sample mode.

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.

14.11. 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 Stream 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.

14.12. 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 Stream 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 Stream truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

In Cribl Stream 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 Stream 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.

14.13. 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.

14.14. 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 Stream.
   You always have the option to upload the file to Cribl Stream'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 Worker Node(s). In a distributed deployment, we recommend having the file on the Leader and all Worker 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 Worker Nodes. However, if you've uploaded `.mmdb` files via Cribl Stream's Lookups Library UI, just click this combo box to display and select them on a drop-down list. Your selection here will handle the path management automatically.

IP field: Field name in which to find an IP to look up. Can be nested. Defaults to `ip`.

Result field : Field name in which to store the GeoIP lookup results. Defaults to `geoip`.

Examples

Assume that you are receiving SMTP logs, and need to see geolocation information associated with IPs using the SMTP service.

Here's a sample of our data, from IPSwitch IMail Server logs:

```
03:19 03:22 SMTPD(00180250) [192.168.1.131] connect 74.136.132.88 port 2539 03:19
03:22 SMTPD(00180250) [74.136.132.88] EHLO msnbc.com 03:19 03:22 SMTPD(00180250)
[74.136.132.88] MAIL FROM:<info-jjgcdshx@test.us> 03:19 03:22 SMTPD(00180250)
[74.136.132.88] RCPT To:<user@domain.com>
```

In this example, we'll chain together three Functions. First, we'll use a Regex Extract Function to isolate the host's IP. Next, we'll use the GeoIP Function to look up the extracted IP against our geoIP database, placing the returned info into a new `__geoip` field. Finally we'll use an Eval Function to parse that field's city, state, country, ZIP, latitude, and longitude.

Function 1 – Regex Extract

Regex: `\[(?<ip>\S+)\]` Source field: `_raw` Result: `74.136.132.88`

Function 2 – GeoIP

Event's IP field: `ip` Result field: `__geoip`

Function 3 – Eval

Name	Value Expression
`City`	`__geoip.city.names.en`
`Country`	`__geoip.country.names.en`
`Zip`	`__geoip.postal.code`
`Lat`	`__geoip.location.latitude`
`Long`	`__geoip.location.longitude`

In the Eval Function’s **Remove fields** setting, you could specify the `__geoip` field for removal, if desired. However, its `__` prefix makes it an internal field anyway.

For a hosted tutorial on applying the GeoIP Function, see Cribl’s GeoIP and Threat Feed Enrichment Sandbox.

14.15. 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 Stream's UI by selecting **Knowledge** > Grok Patterns.
Pattern files are located at: `$CRIBL_HOME/(default|local)/cribl/grok-patterns/`

Example

Example event:

```
{"_raw": "2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message"}
```

Pattern: `%{TIMESTAMP_ISO8601:event_time} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}` **Source Field**: `_raw`

Event after extraction:

```
{"_raw": "2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message",
  "_time": 1600226442.045,
  "event_time": "2020-09-16T04:20:42.45+01:00",
  "log_level": "DEBUG",
  "log_message": "This is a sample debug log message",
}
```

Note the new fields added to the event: `event_time`, `log_level`, and `log_message`.

References

- Syntax for a Grok pattern is `%{PATTERN_NAME:FIELD_NAME}`. E.g.: `%{IP:client} %{WORD:method}`.
- Useful link for creating and testing Grok patterns: http://grokconstructor.appspot.com
- Additional patterns are available here: https://github.com/logstash-plugins/logstash-patterns-core/tree/main/patterns

14.16. 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":"I

Event 2:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"(

Event 3:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"I

Event 4:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name":"I
```

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**

14.17. 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 Stream'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 Stream 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 modes**, 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 Stream'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 Stream's UI, a distributed Cribl Stream deployment will override this setting as necessary – skipping checks to prevent conflicts that could trigger skipped lookups. These restrictions do **not** apply to single-instance deployments.

Ignore case: Ignore case when performing **Match mode: Exact** lookups. Defaults to `No`.

Add to raw event: Whether to append the looked-up values to the `_raw` field, as key=value pairs. Defaults to `No`.

Examples

Example 1: Regex Lookups

Assign a `sourcetype` field to events if their `_raw` field matches a particular regex.

paloalto.csv

```
regex,sourcetype
"^[^,]+,[^,]+,[^,]+,THREAT",pan:threat
"^[^,]+,[^,]+,[^,]+,TRAFFIC",pan:traffic
"^[^,]+,[^,]+,[^,]+,SYSTEM",pan:system
```

Match mode: Regex

Match type: First match

When using the Lookup Function with Regex and `First match`, ensure that your lookup file contains no empty lines - not even at the bottom. Any empty row will cause the function to always return `true`.

Lookup field name in event: `_raw`

Corresponding field name in lookup: `regex`

Events before and after

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/(


### AFTER:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/
  "sourcetype": "pan:traffic"
  }
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/(
  "sourcetype": "pan:threat"
  }
```

Example 2: CIDR Lookups

Assign a `location` field to events if their `destination_ip` field matches a particular CIDR range.

paloaltoips.csv

```
range,location
10.0.0.0/24,San Francisco
10.0.0.0/16,California
10.0.0.0/8,US
```

Match mode: CIDR

Match type: See options below

Lookup field name in event: `destination_ip`

Corresponding field name in lookup: `range`

In **Match mode: CIDR** with **Match type: Most specific**, the lookup will implicitly search for matches from most specific to least specific. There is no need to pre-sort data.

Note that **Match mode: CIDR** with **Match type: First Match** is likely the most performant with large lookups. This can be used as an alternative to **Most specific**, if the file is sorted with the most specific/relevant entries first. This mode still performs a table scan, top to bottom.

Events before and after

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,201
  "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,2049,201
  "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,2049,201
  "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,2049,201
  "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.

14.18. 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.

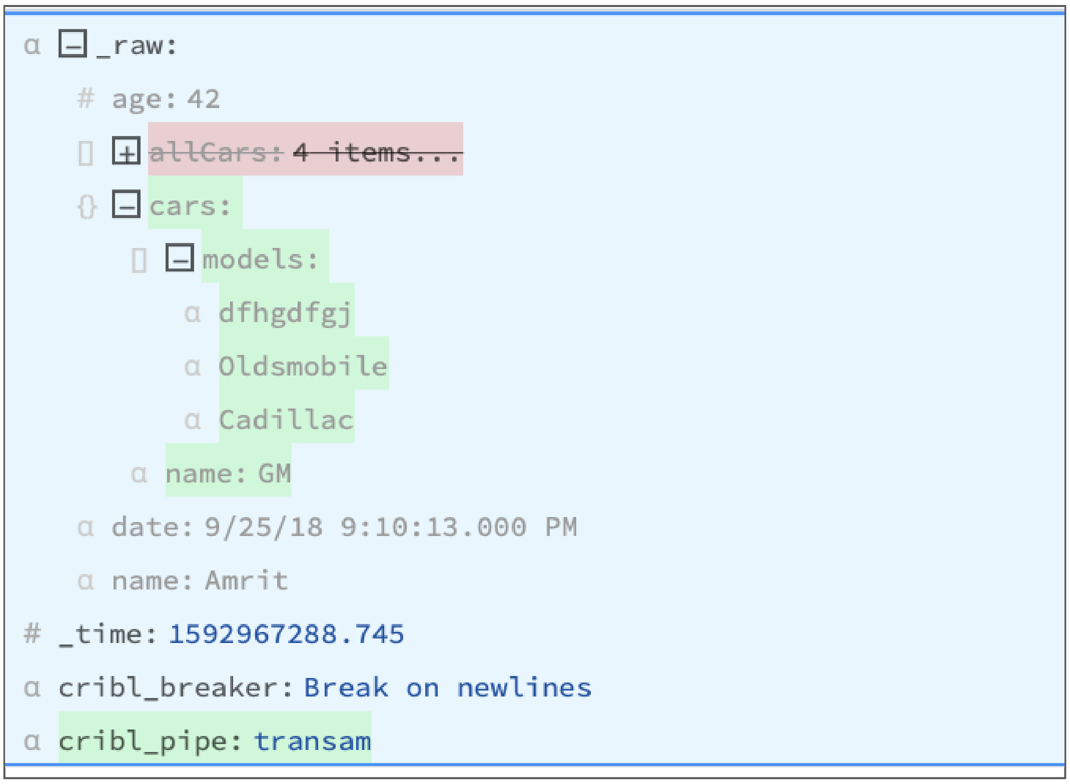
For example, suppose a **Replace expression** that generates a random integer is applied to a field named `bytes` and produces the number `8403`. Cribl Stream will output an event where the value of the `bytes` field is the **string** `8403`.

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:

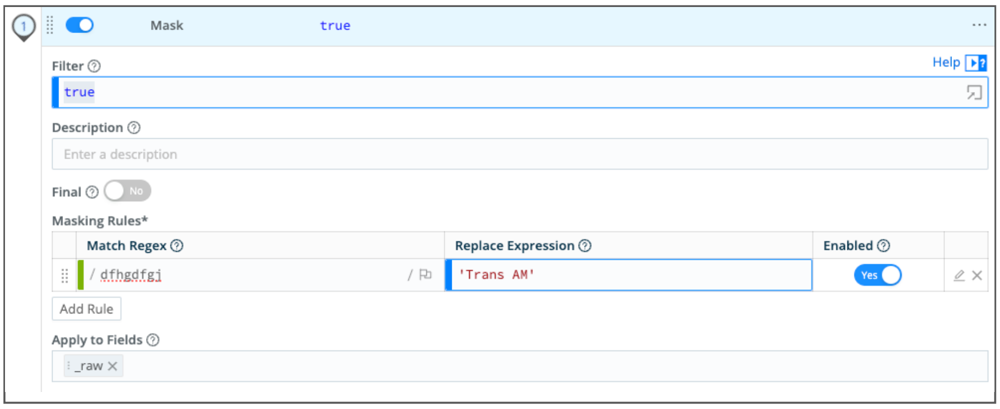


```
α ⊟ _raw:
    # age: 42
    [] ⊞ ~~allCars: 4 items...~~
    {} ⊟ cars:
        [] ⊟ models:
            α dfhgdfgj
            α Oldsmobile
            α Cadillac
        α name: GM
    α date: 9/25/18 9:10:13.000 PM
    α name: Amrit
# _time: 1592967288.745
α cribl_breaker: Break on newlines
α cribl_pipe: transam
```

Event before masking

Configure the Mask Function > Masking Rules as follows:

Match Regex: `dfhgdfgj` Replace Expression: `Trans AM`



Mask true

Filter: `true`

Description: Enter a description

Final: No

Masking Rules*

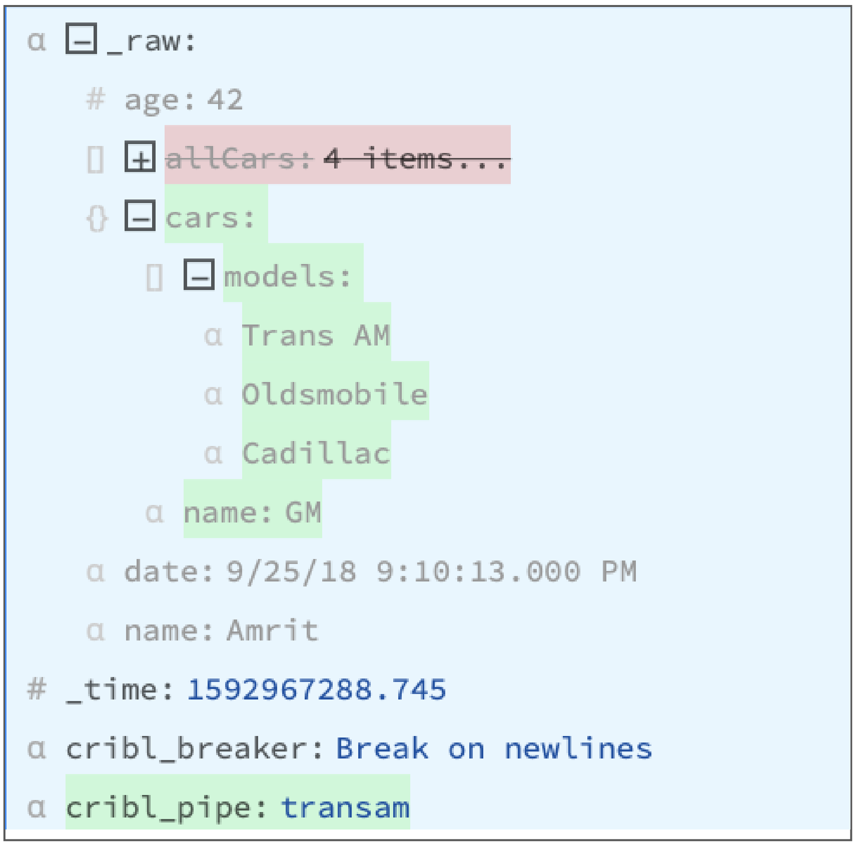
Match Regex	Replace Expression	Enabled
/dfhgdfgj/	'Trans AM'	Yes

Add Rule

Apply to Fields: _raw

Mask Function configuration

Result: Vroom vroom!

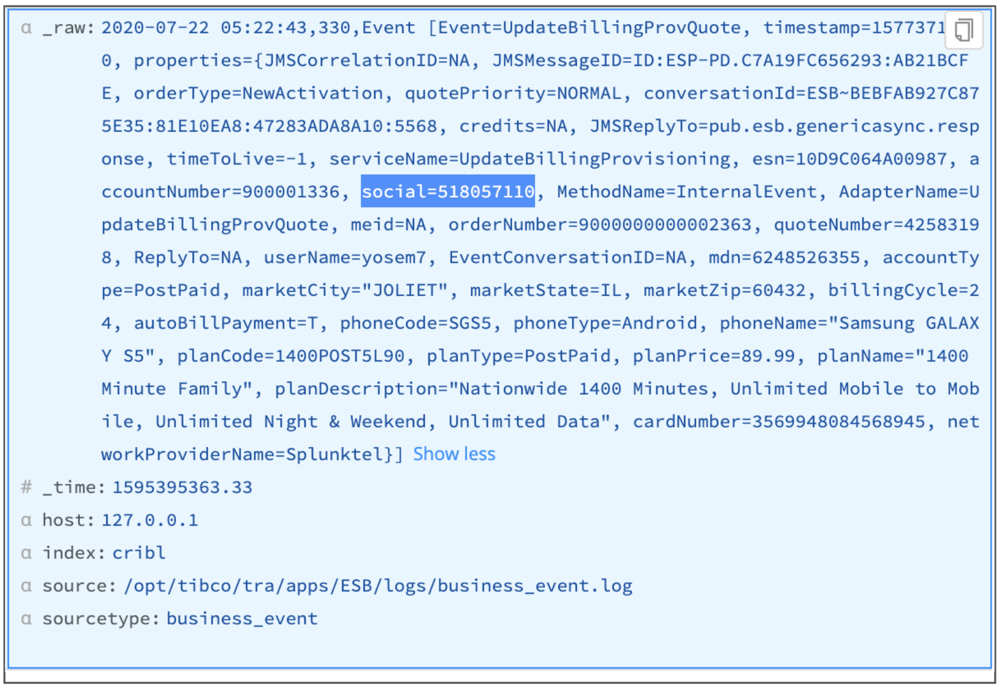


```
_raw:
  age: 42
  ~~allCars: 4 items...~~
  cars:
    models:
      Trans AM
      Oldsmobile
      Cadillac
    name: GM
  date: 9/25/18 9:10:13.000 PM
  name: Amrit
_time: 1592967288.745
cribl_breaker: Break on newlines
cribl_pipe: transam
```

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=#########`.



```
_raw: 2020-07-22 05:22:43,330,Event [Event=UpdateBillingProvQuote, timestamp=1577371
      0, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.C7A19FC656293:AB21BCF
      E, orderType=NewActivation, quotePriority=NORMAL, conversationId=ESB~BEBFAB927C87
      5E35:81E10EA8:47283ADA8A10:5568, credits=NA, JMSReplyTo=pub.esb.genericasync.resp
      onse, timeToLive=-1, serviceName=UpdateBillingProvisioning, esn=10D9C064A00987, a
      ccountNumber=900001336, social=518057110, MethodName=InternalEvent, AdapterName=U
      pdateBillingProvQuote, meid=NA, orderNumber=9000000000002363, quoteNumber=4258319
      8, ReplyTo=NA, userName=yosem7, EventConversationID=NA, mdn=6248526355, accountTy
      pe=PostPaid, marketCity="JOLIET", marketState=IL, marketZip=60432, billingCycle=2
      4, autoBillPayment=T, phoneCode=SGS5, phoneType=Android, phoneName="Samsung GALAX
      Y S5", planCode=1400POST5L90, planType=PostPaid, planPrice=89.99, planName="1400
      Minute Family", planDescription="Nationwide 1400 Minutes, Unlimited Mobile to Mob
      ile, Unlimited Night & Weekend, Unlimited Data", cardNumber=3569948084568945, net
      workProviderName=Splunktel}] Show less
_time: 1595395363.33
host: 127.0.0.1
index: cribl
source: /opt/tibco/tra/apps/ESB/logs/business_event.log
sourcetype: business_event
```

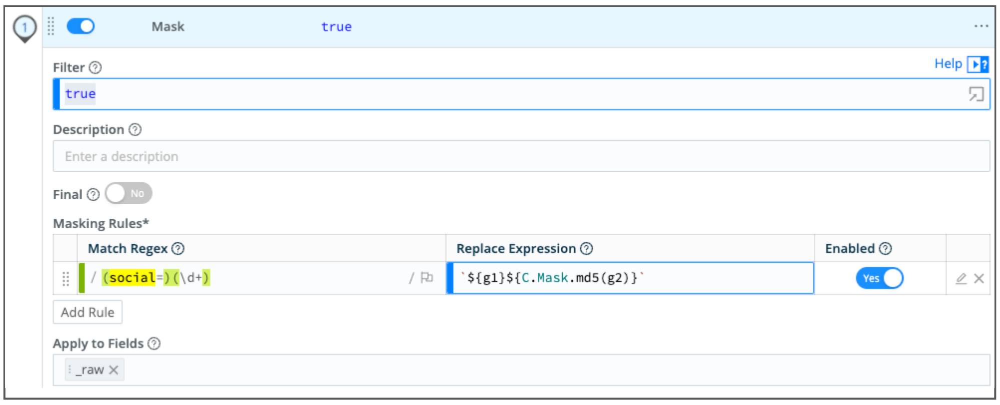
Event with unredacted SSNs

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.

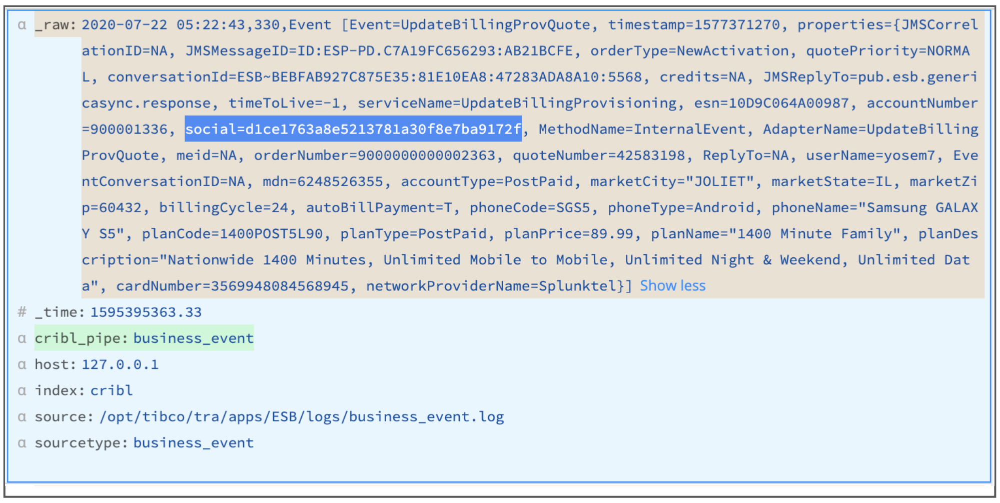
In this example, `social=` is assigned to capture group g1 for later reference. The value of `social=` will be hashed by referencing it as g2 in the md5 function. If we didn't make `social=` its own capture group (or specified `social=` as a literal in the Replace Expression) then we cannot reference 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                    true
Filter: true
Description: Enter a description
Final: No
Masking Rules*
Match Regex: /(social=)(\d+)/
Replace Expression: `${g1}${C.Mask.md5(g2)}`
Enabled: Yes
Apply to Fields: _raw
```

Mask Function configuration

Result: The sensitive values are replaced by their md5 hashes.



```
_raw: 2020-07-22 05:22:43,330,Event [Event=UpdateBillingProvQuote, timestamp=1577371270, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.C7A19FC656293:AB21BCFE, orderType=NewActivation, quotePriority=NORMAL, conversationId=ESB~BEBFAB927C875E35:81E10EA8:47283ADA8A10:5568, credits=NA, JMSReplyTo=pub.esb.genericasync.response, timeToLive=-1, serviceName=UpdateBillingProvisioning, esn=10D9C064A00987, accountNumber=900001336, social=d1ce1763a8e5213781a30f8e7ba9172f, MethodName=InternalEvent, AdapterName=UpdateBillingProvQuote, meid=NA, orderNumber=9000000000002363, quoteNumber=42583198, ReplyTo=NA, userName=yosem7, EventConversationID=NA, mdn=6248526355, accountType=PostPaid, marketCity="JOLIET", marketState=IL, marketZip=60432, billingCycle=24, autoBillPayment=T, phoneCode=SGS5, phoneType=Android, phoneName="Samsung GALAXY S5", planCode=1400POST5L90, planType=PostPaid, planPrice=89.99, planName="1400 Minute Family", planDescription="Nationwide 1400 Minutes, Unlimited Mobile to Mobile, Unlimited Night & Weekend, Unlimited Data", cardNumber=3569948084568945, networkProviderName=Splunktel}] Show less
# _time: 1595395363.33
cribl_pipe: business_event
host: 127.0.0.1
index: cribl
source: /opt/tibco/tra/apps/ESB/logs/business_event.log
sourcetype: business_event
```

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
```

14.19. 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 Stream or downstream services:

```
03:19 03:22 SMTPD (00180250) [209.221.59.70] C:\IMail\spool\D28de0018025017cd.SMD
3827
```

Again, we can accomplish this with two Functions:

1. A Regex Extract Function. To capture a substring of digits that follows six other substrings (all separated by white space), we set the **Regex** field to: `\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+\S+\s+(?<bytes>\d+)`
2. Then use Numerify.

14.20. OTLP Metrics

The OTLP Metrics Function transforms dimensional metrics events into the OTLP (OpenTelemetry Protocol) format. This Function can be used to send metric events (events that contain the internal field `__criblMetrics`) to any OpenTelemetry Metrics-capable destinations.

For detailed insights into events that passed through the Function but were dropped, set the logging level for channel `func:otlp_metrics` to `debug`, and there will be a stats log message output once per minute with the relevant information.

This Function will delete the internal attribute `__criblMetrics`, making events only suitable for destinations that use the OTLP format.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. The default `true` setting passes all events through the Function.

Description: Simple description of this Function's purpose in this Pipeline. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Drop non-metric events: Whether or not to drop any non-metric events.

Basic Example

Filter: `__inputId=='prometheus_rw:prom_rw_in'`

This will grab all events from a Prometheus Remote Write Source and convert the metrics to OTLP format.

Ensure that the internal attribute `__criblMetrics` does not contain `null` or blank values in any of the child object arrays.

Sample event:

```
{
  "__criblEventType": "event",
  "__ctrlFields": [],
  "__final": false,
  "__cloneCount": 0,
  "_time": 1700236380.37,
  "_value": 123.456,
  "_metric": "cribl_logstream_health_outputs",
  "_metric_type": "gauge",
  "cribl_host": "P4MD6R-Pv8x",
  "cribl_wp": "w0",
  "event_host": "P4MD6R-Pv8x",
  "event_source": "cribl",
  "host": "P4MD6R-Pv8x",
  "output": "open_telemetry:otel_webhook_dest",
  "source": "cribl",
  "__criblMetrics": [
    {
      "nameExpr": ["_metric"],
      "values": ["_value"],
      "types": ["gauge"],
      "dims": [
        "cribl_host",
        "cribl_wp",
        "event_host",
        "event_source",
        "host",
        "output",
        "source"
      ]
    }
  ],
  "__offset": 8099,
  "__bytes": 228,
  "__srcIpPort": "127.0.0.1:50868",
  "__inputId": "prometheus_rw:prom_rw_in"
}
```

The same sample event after conversion by the OTLP Metrics Function:

```
{
  "_time": 1700236380.37,
  "instrumentation_library_metrics": [
    {
      "schema_url": "",
      "metrics": [
        {
          "__type": "gauge",
          "gauge": {
            "data_points": [
              {
                "attributes": [
                  {
                    key: "cribl_host",
                    value: {
                      string_value: "P4MD6R-Pv8x",
                    }
                  },
                  {
                    key: "cribl_wp",
                    value: {
                      string_value: "w0",
                    }
                  },
                  {
                    key: "event_host",
                    value: {
                      string_value: "P4MD6R-Pv8x",
                    }
                  },
                  {
                    key: "event_source",
                    value: {
                      string_value: "cribl",
                    }
                  },
                  {
                    key: "host",
                    value: {
                      string_value: "P4MD6R-Pv8x",
                    }
                  },
                  {
                    key: "output",
                    value: {
                      string_value: "open_telemetry:otel_webhook_dest",
                    }
                  },
                  {
                    key: "source",
                    value: {
                      string_value: "cribl"
                    }
                  },
                ],
                "start_time_unix_nano": 1700236380370000000,
                "time_unix_nano": 1700236380370000000,
                "as_double": 123.456
              }
            ]
          },
          "name": "cribl_logstream_health_outputs"
        }
      ]
    }
  ]
}
```

14.21. 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

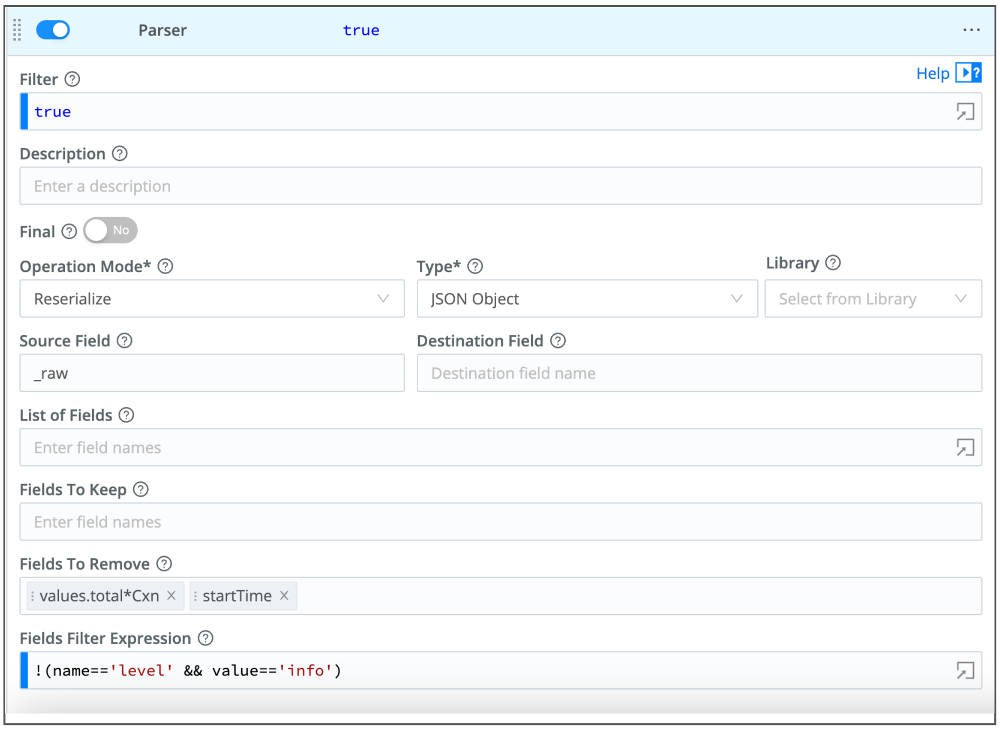
Examples

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 `true`

Filter: `true`

Description: Enter a description

Final: No

Operation Mode*: Reserialize

Type*: JSON Object

Library: Select from Library

Source Field: _raw

Destination Field: Destination field name

List of Fields: Enter field names

Fields To Keep: Enter field names

Fields To Remove: values.total*Cxn, startTime

Fields Filter Expression:

```
!(name=='level' && value=='info')
```

Parser Function initial configuration

First, set the **Type** to `Key=Value Pairs`.

Scenario A

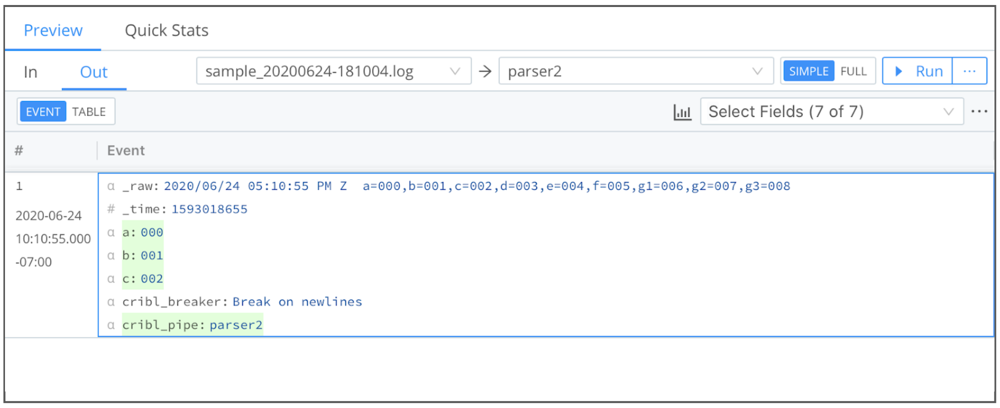
Keep fields `a, b, c`. Drop the rest.

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`


Preview Quick Stats
In Out sample_20200624-181004.log → parser2 SIMPLE FULL Run
EVENT TABLE Select Fields (7 of 7)

```
#  Event
1  _raw: 2020/06/24 05:10:55 PM Z  a=000,b=001,c=002,d=003,e=004,f=005,g1=006,g2=007,g3=008
2020-06-24  _time: 1593018655
10:10:55.000  a: 000
-07:00  b: 001
   c: 002
   cribl_breaker: Break on newlines
   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`

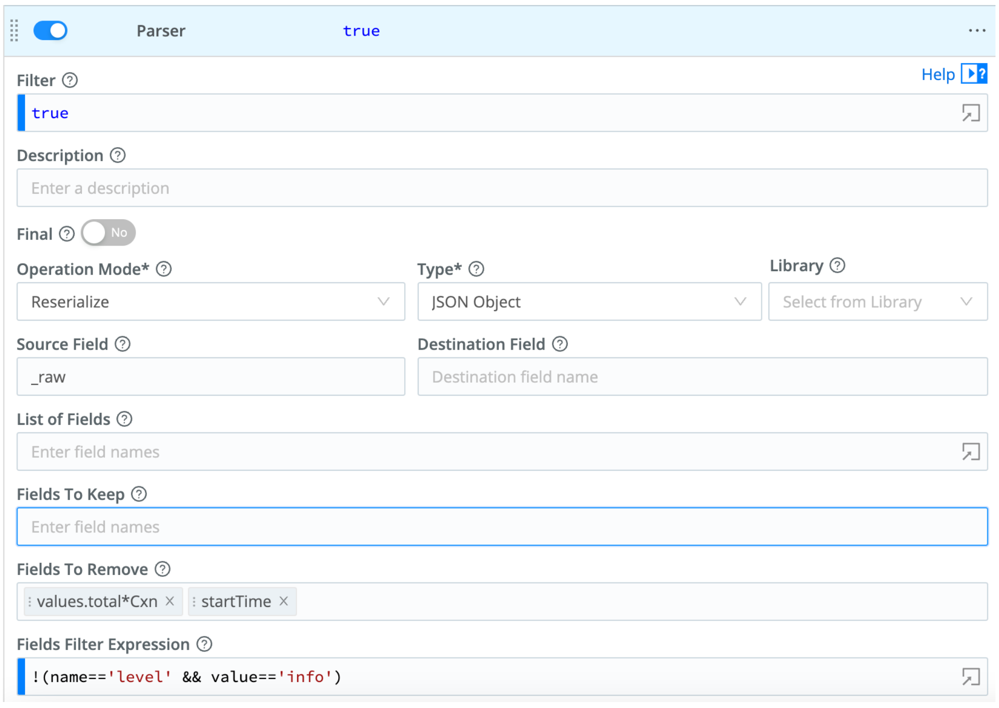
The `index` refers to the location of a field in the array of all fields extracted by **this** Parser. It is zero-based. In the case above, `g2` and `g3` have `index` values of `7` and `8`, respectively.

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`.

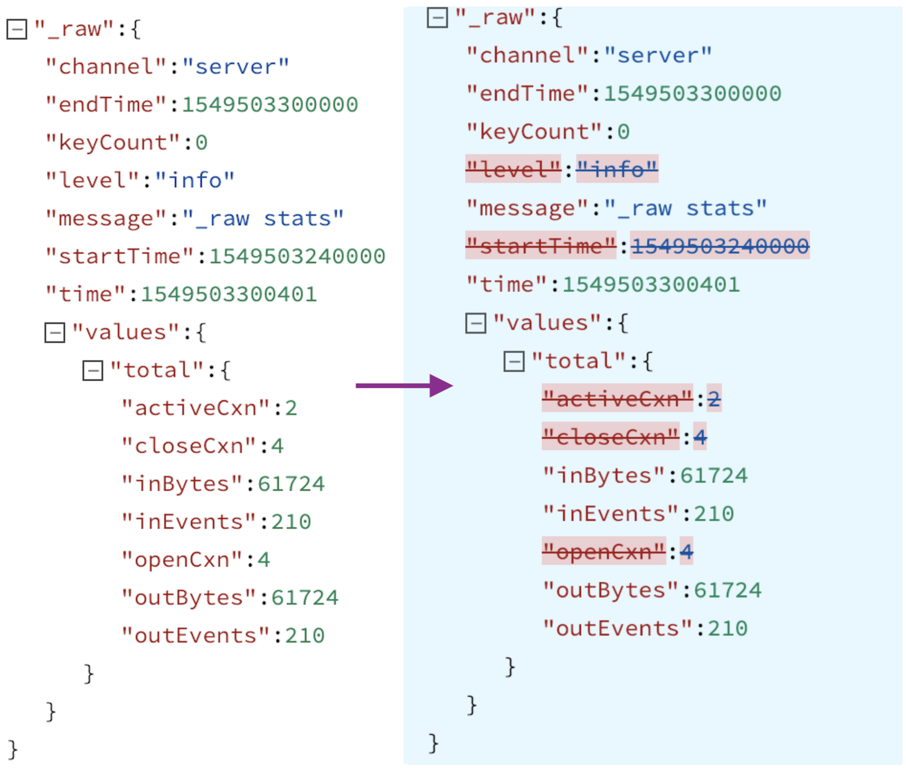
Parser Function configuration:



Parser	true
Filter	true
Description	Enter a description
Final	No
Operation Mode*	Reserialize
Type*	JSON Object
Library	Select from Library
Source Field	_raw
Destination Field	Destination field name
List of Fields	Enter field names
Fields To Keep	Enter field names
Fields To Remove	values.total*Cxn, startTime
Fields Filter Expression	!(name=='level' && value=='info')

Parser Function configuration for Example 2

JSON event after being processed by the Function:



```
"_raw":{
  "channel":"server"
  "endTime":1549503300000
  "keyCount":0
  "level":"info"
  "message":"_raw stats"
  "startTime":1549503240000
  "time":1549503300401
  "values":{
    "total":{
      "activeCxn":2
      "closeCxn":4
      "inBytes":61724
      "inEvents":210
      "openCxn":4
      "outBytes":61724
      "outEvents":210
    }
  }
}
```

```
"_raw":{
  "channel":"server"
  "endTime":1549503300000
  "keyCount":0
  ~~"level":"info"~~
  "message":"_raw stats"
  ~~"startTime":1549503240000~~
  "time":1549503300401
  "values":{
    "total":{
      ~~"activeCxn":2~~
      ~~"closeCxn":4~~
      "inBytes":61724
      "inEvents":210
      ~~"openCxn":4~~
      "outBytes":61724
      "outEvents":210
    }
  }
}
```

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:

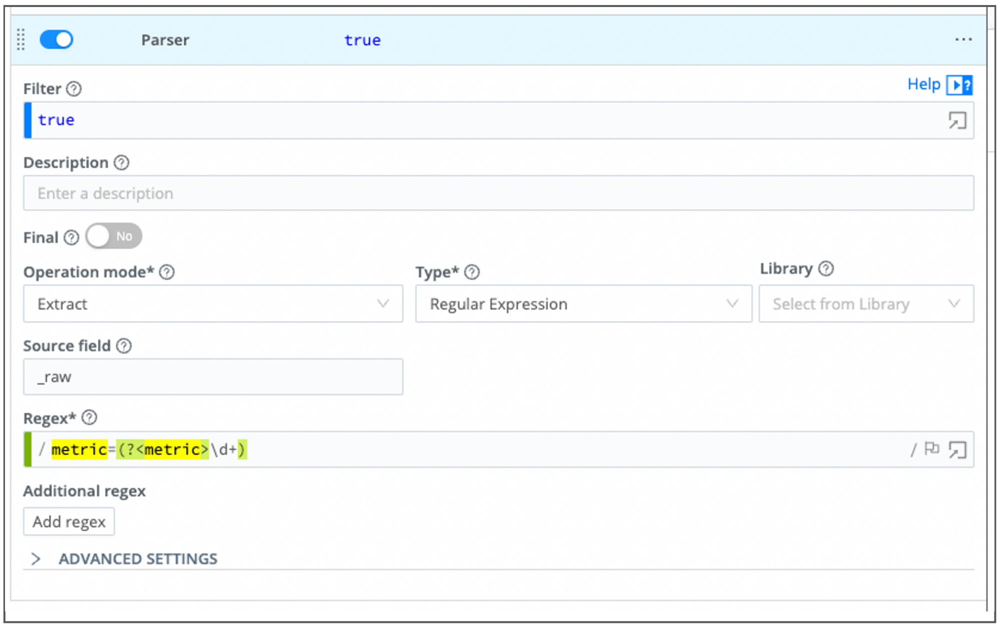
- Operation mode: `Serialize`
- Source field: [empty]
- Destination field: `bar`
- Type: `JSON`
- List of fields: [empty]
- Fields to keep: `a`, `b`, `c`, `d`

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 true

Filter: true

Description: Enter a description

Final: No

Operation mode*: Extract

Type*: Regular Expression

Library: Select from Library

Source field: _raw

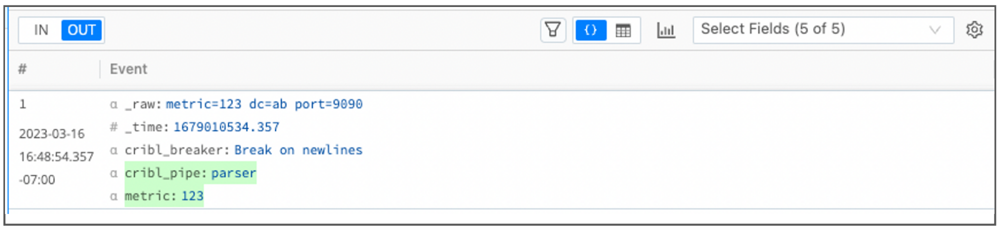
Regex*: /metric=(?<metric>\d+)/

Additional regex: Add regex

ADVANCED SETTINGS

Parser Function configuration for Example 4

The output will look like this:



```
#                          Event
1                          a _raw: metric=123 dc=ab port=9090
2023-03-16                 # _time: 1679010534.357
16:48:54.357               a cribl_breaker: Break on newlines
-07:00                     a cribl_pipe: parser
                           a metric: 123
```

Parser Function regex output

Example 5

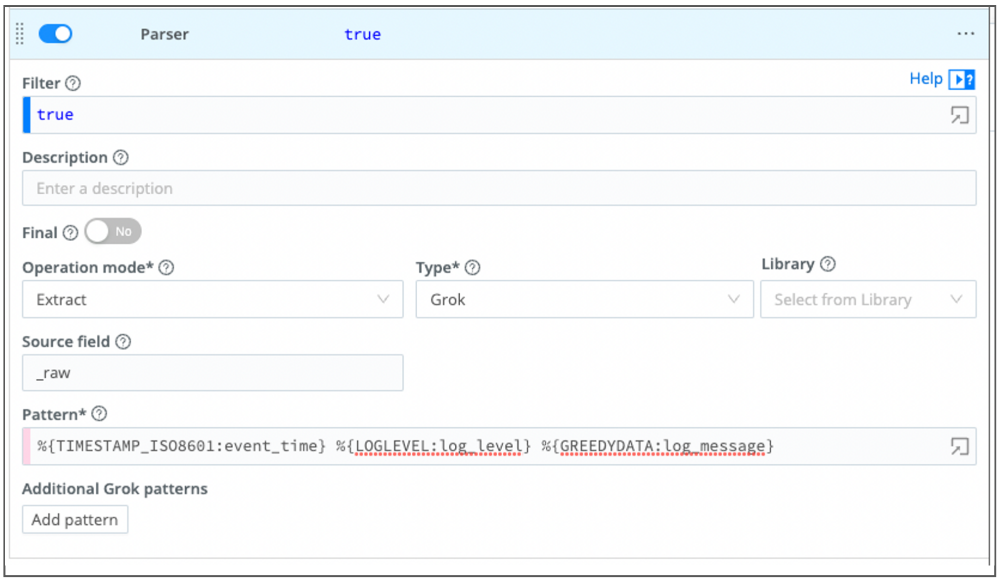
Assume we have an event that looks like this:

```
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 true

Filter: true

Description: Enter a description

Final: No

Operation mode*: Extract

Type*: Grok

Library: Select from Library

Source field: _raw

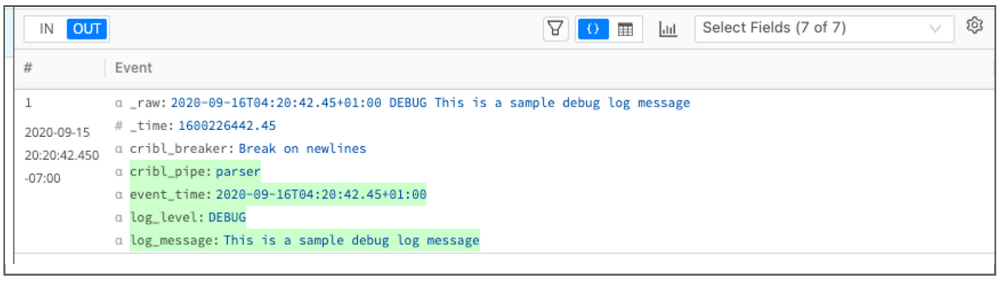
Pattern*:
```
%{TIMESTAMP_ISO8601:event_time} %{LOGLEVEL:log_level} %{GREEDYDATA:log_message}
```

Additional Grok patterns

Add pattern

Parser Function configuration for Example 5

The output will look like this:



```
1                       α _raw: 2020-09-16T04:20:42.45+01:00 DEBUG This is a sample debug log message
2020-09-15              # _time: 1600226442.45
20:20:42.450            α cribl_breaker: Break on newlines
-07:00                  α cribl_pipe: parser
                        α event_time: 2020-09-16T04:20:42.45+01:00
                        α log_level: DEBUG
                        α log_message: This is a sample debug log message
```

Parser Function grok output

14.22. 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 Stream will remove that metric from the event.

Dimensions

Add Dimensions: Optional list of dimensions to include in events. Supports wildcards. If you don't specify metrics, values will be appended to every metric found in the event. When you add a new metric, dimensions will be present only in those new metrics. Defaults to `!_* *`.

Remove Dimensions: Optional list of dimensions to associate with every extracted metric value. Leave blank if this Function processes output from the Aggregations Function as dimensions will be automatically discovered.

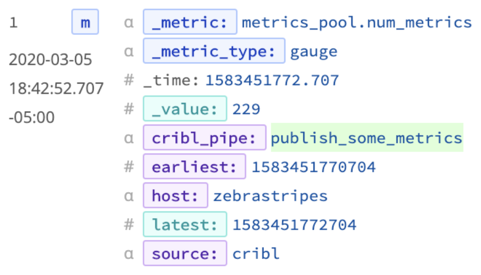
Overwrite: If set to `Yes`, overwrite previous metric specs. Otherwise, append. Defaults to `No`.

The **Add Dimensions** and **Remove Dimensions** fields support wildcards and negated terms. When you use negated terms, the list is order-sensitive. E.g., `!foobar` before `foo*` means "All fields that start with `foo`, except `foobar`." However, `!foo*` before `*` means "All fields, except for those that start with `foo`."

Cribl Stream 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:



```
1            m   α _metric: metrics_pool.num_metrics
                 α _metric_type: gauge
2020-03-05       # _time: 1583451772.707
18:42:52.707     # _value: 229
-05:00           α cribl_pipe: publish_some_metrics
                 # earliest: 1583451770704
                 α host: zebrastripes
                 # latest: 1583451772704
                 α source: cribl
```

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**

Event Field Name	Metric Name Expression	Metric Type
`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:
metric.10.10.12.192.bytes_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:e
metric.10.10.12.192.pkts_sent:1|c|#destination_ip:160.177.222.249,inbound_interface:et
metric.10.10.12.192.pkts_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:et
```

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.

14.23. 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 Stream Collectors (for example, a REST Collector) to retrieve reference data from desired endpoints, and then use this Function to store the data on Redis and retrieve it to enrich your data. By default, Cribl Stream does not cache the data returned from this Redis Function, but you can enable client-side caching.

Your Redis Function can interact with three kinds of Redis instances: Standalone (the default); Redis Cluster, which scales horizontally; or Redis Sentinel, a distributed system that provides high availability.

Managing Redis Connections

When using Redis Functions, you must consider how to avoid creating large numbers of TCP connections to the Redis server (which we'll call "Redis connections"). To manage Redis connections, you start by calculating how many Redis connections your Redis Functions, other functions, and Pipelines will create using default settings. Then, to keep the overall number of connections reasonable, you configure Cribl Stream to **reuse** Redis connections.

What causes the number of Redis connections to grow large? By default, Cribl Stream creates a separate TCP connection to Redis for every Redis Function and every **reference to** a Redis Function. This means that certain usage patterns will increase the number of Redis Connections, including: sending the output of a Chain Function to a regular Pipeline that includes a Redis Function; using a Redis Function in a Source's pre-processing Pipeline; and, using a Redis Function in a Destination's post-processing Pipeline.

How can you control the number of Redis connections created? You use the **Reuse Redis connections** control. Then, when a new Redis connection is needed, **and** its connection properties are identical to those of an existing Redis connection, Cribl Stream will try to reuse the existing connection instead of creating a new one. The example below illustrates how powerfully this can affect the demands Cribl Stream places on a Redis server.

What are the connection properties that must be identical for reuse to be possible? For the different kinds of Redis instances, the relevant connection properties are, respectively:

- Standalone: username, password, and URL.
- Cluster: username, password, and all root node hosts and ports.
- Redis Sentinel: username, password, master name, and all root node hosts and ports.

In managing Redis connections, the factors you must account for include: what kind of Redis instance you're connecting to (Standalone, Cluster, or Redis Sentinel); the size of the Redis server; the number of connections the Redis server is configured to allow; the number of other clients using the same Redis server; and so on.

Where to configure **Reuse Redis connections** in the UI depends on how you have deployed Cribl Stream:

- Distributed deployment: Navigate to **Limits** at the Worker Group level.
- Single-instance deployment: Navigate to **Limits** at the global (**General Settings**) level.

Example: Managing Redis Connections

Referencing a Redis Function from a Pipeline and/or another Function can dramatically multiply the number of Redis connections created, unless we mitigate this effect by applying the **Reuse Redis connections** control. Let's see how this plays out in a detailed example. Suppose we do the following:

1. Create a Pipeline (we'll call it "Pipeline Zero") that contains 10 Redis Functions that connect to the same Redis server, with identical connection properties. Cribl Stream will create 10 TCP connections to the Redis server ("Redis connections").
2. Reference Pipeline Zero in a Chain Function that's in another Pipeline. This creates 10 more connections, for a total of 20 Redis connections.
3. Reference Pipeline Zero in an additional, separate Chain Function that's in yet another Pipeline, creating 10 more connections, for a total of 30 Redis connections.
4. Use Pipeline Zero as a pre-processing Pipeline for a Source, creating 10 more connections, for a total of 40 Redis connections.
5. Use Pipeline Zero as a post-processing Pipeline for a Destination, creating 10 more connections. Now we have a grand total of 50 Redis connections.

Now consider the effect of these Pipelines within Worker Nodes and their Worker Processes. For a typical deployment scenario with 10 Worker Nodes, each of which have 10 Worker Processes, the calculation looks like this:

- 50 connections times 10 Worker Processes = 500 connections, times 10 Worker Nodes = 5000 connections.

A single Redis server can be hard-pressed to handle so many connections. Apart from that, the number of connections is unnecessarily high for the expected throughput. Let's use **Reuse Redis connections** to reduce the number of Redis connections.

Enable **Reuse Redis connections** and set **Max connections** to `4`. Now revise the calculation:

- 4 connections times 10 Worker Processes = 40 connections, times 10 Worker Nodes = 400 connections.

This reduces the number of connections by better than a factor of 10, making our deployment much less likely to overwhelm the Redis server.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Result field: Name of the field in which to store the returned value. (Leave empty to discard the returned value.)

Command: Redis command to perform. Required. (A complete list of Redis commands is at: https://redis.io/commands.)

Key: A JavaScript expression to compute the value of the key to operate on. Can also be a constant such as `username`. This is a required field. Click the icon at right to open a validation modal.

Args: A JavaScript expression to compute arguments to the operation. Click the icon at right to open a validation modal. When the expression you enter is a plain value or an array, it resembles the options and arguments of a Redis command in the Redis CLI.

For example:

- Suppose you want to `SET` a key named `_mascot`. And, of the options that the `SET` command takes, you want to use two: `GET` and `EX`. The first, `GET`, takes no arguments; the second, `EX`, takes an integer in seconds.
- You want to set `_mascot` to the value `goat`, and `EX` to 5 seconds. Here is the array you'd enter in the Args column:
  `['goat', 'GET','EX',5 ]`

Deployment Type

From the drop-down, select `Standalone`, `Cluster`, or `Sentinel`. Some of these options display additional controls below. Hostnames must be JavaScript expressions (which can evaluate to a constant value), enclosed in quotes or backticks. They can be evaluated only at init time.

Standalone

This is the default deployment type. Enter a **Redis URL** to connect to. The format is:
`[redis[s]:]//[[user][:password@]][host][:port][/db-number][?db=db-number[&password=bar[&option=value]]]`

Example:
`redis://user:secret@localhost:6379/0?foo=bar&qux=baz`

Example with no `user` specified:
`redis://secret@localhost:6379/0?foo=bar&qux=baz`

Standalone with TLS

To enable TLS, specify a Redis URL that starts with `rediss://` rather than `redis://`. This URL reveals the TLS section of the config UI.

Example:
`rediss://user:secret@localhost:6379/0?foo=bar&qux=baz`

Example with no `user` specified:
`rediss://secret@localhost:6379/0?foo=bar&qux=baz`

Cluster

Specify **Root Nodes** that the cluster will connect to. A single node is sufficient, but Cribl recommends listing two or more. Each of the **Root Nodes** consists of a **Hostname** and a **Port**.

> In multi-key commands, ensure that all keys in the command are in the same hash slot. Otherwise, the command will fail. As a workaround, you can split the multi-key command into separate commands, where each command operates only on keys from a single hash slot.

Sentinel

Specify a **Master Group Name**, plus every **Sentinel** in your Redis deployment. Each **Sentinel** consists of a **Hostname** and a **Port**.

Authentication Method

Use the **Authentication method** buttons to select one of the following options, some of which display additional controls below.

- **None**: Select this option where authentication either is not required, or is provided in the URL.
- **Basic**: This displays **Username** and **Password** fields for you to enter your Redis credentials.
- **User Secret**: This option exposes a drop-down in which you can select a stored text secret that references a Redis username and password, as described above. A **Create** link is available to store a new, reusable secret.
- **Admin Secret**: This option exposes a drop-down in which you can select a stored text secret that references a Redis admin password. A **Create** link is available to store a new, reusable secret.

TLS

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (such as the system's CA). Defaults to `Yes`.

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting. The minimum TLS version for Redis is version 6.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Minimum TLS Requirement

The minimum Redis version required for TLS support is version 6.

Advanced Settings

Max blocking time: Maximum amount of time (in seconds) before assuming that Redis is down and passing events through. Defaults to `60` seconds. Use `0` to disable timeouts.

Client-side cache: Toggle to `Yes` to enable. Then navigate to **Settings** > **General Settings** > **Limits** > **Redis Cache** to configure the cache, as described in the next section.

Overall Redis Cache Settings

Together, these settings determine how the Cribl Stream 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 Stream 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 Stream 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 Stream-Redis interactions have, the greater that benefit will be. When your Cribl Stream 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 Stream are writing to your Redis instance. Say one of those applications modifies a key's value. Now the value that Cribl Stream client-side cache has for that key is stale. Cribl Stream 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 Stream 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 Stream. 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 Stream client has requested. Redis only notifies Cribl Stream when the value of one of **those** keys has changed. This is most efficient for Cribl Stream 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 Stream when the value of **any** key has changed. This is most efficient for Redis, but now the the Cribl Stream 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 Stream never sees.

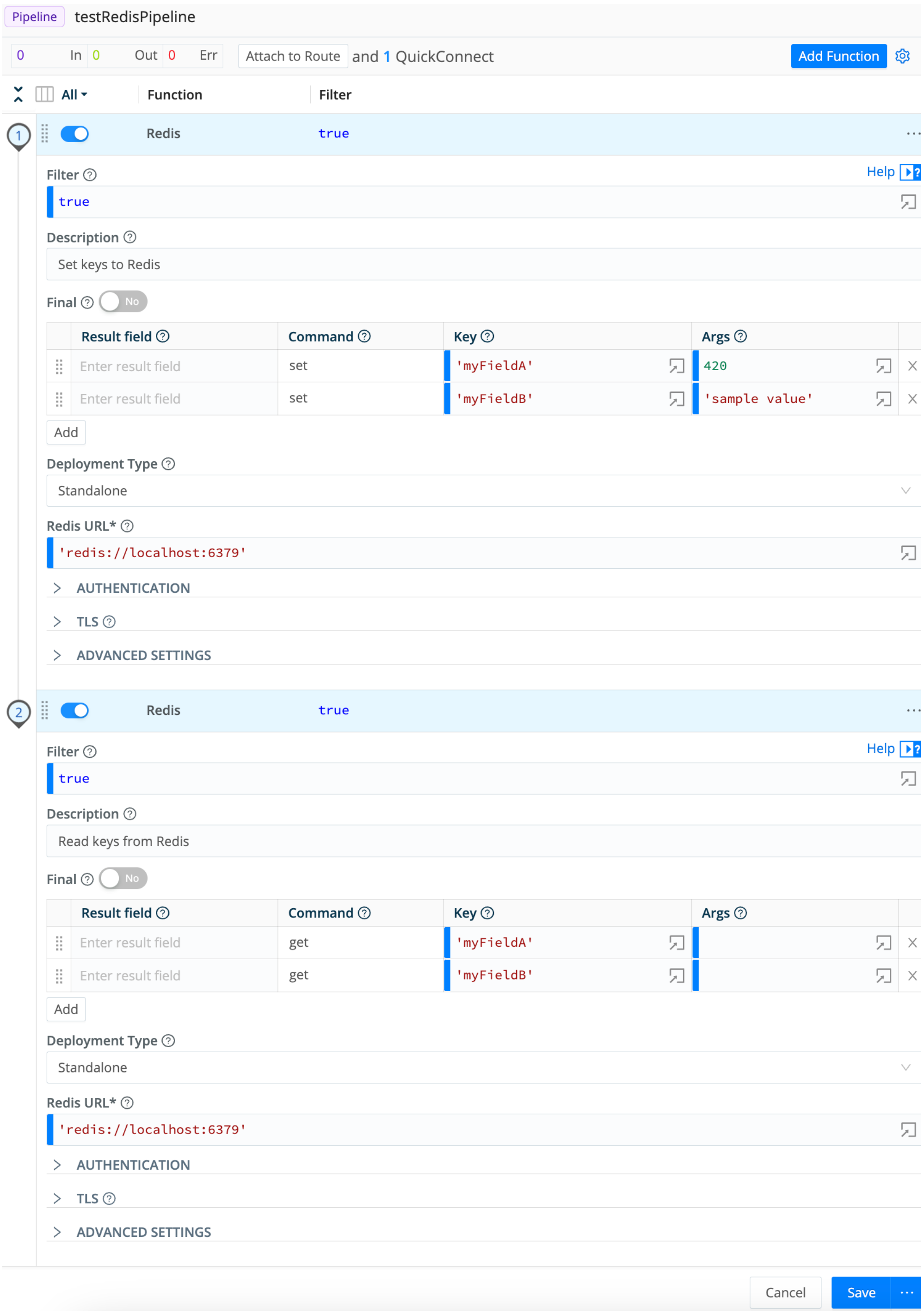
You should select whichever of these mechanisms best suits your environment. Does it make more sense to impose the heavier burden on the Redis server (as in **Default** mode) or on Cribl Stream (as in **Broadcast** mode)?

In both scenarios, for a given key, any write operation from any application (including Cribl Stream) 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 fields**.


Pipeline testRedisPipeline

0 In 0 Out 0 Err | Attach to Route and 1 QuickConnect | Add Function

	Function	Filter
1	Redis	`true`

Filter: `true`

Description: Set keys to Redis

Final: No

Result field	Command	Key	Args
Enter result field	set	`'myFieldA'`	`420`
Enter result field	set	`'myFieldB'`	`'sample value'`

Add

Deployment Type: Standalone

Redis URL*: `'redis://localhost:6379'`

AUTHENTICATION

TLS

ADVANCED SETTINGS

	Function	Filter
2	Redis	`true`

Filter: `true`

Description: Read keys from Redis

Final: No

Result field	Command	Key	Args
Enter result field	get	`'myFieldA'`	
Enter result field	get	`'myFieldB'`	

Add

Deployment Type: Standalone

Redis URL*: `'redis://localhost:6379'`

AUTHENTICATION

TLS

ADVANCED SETTINGS

Cancel | Save

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

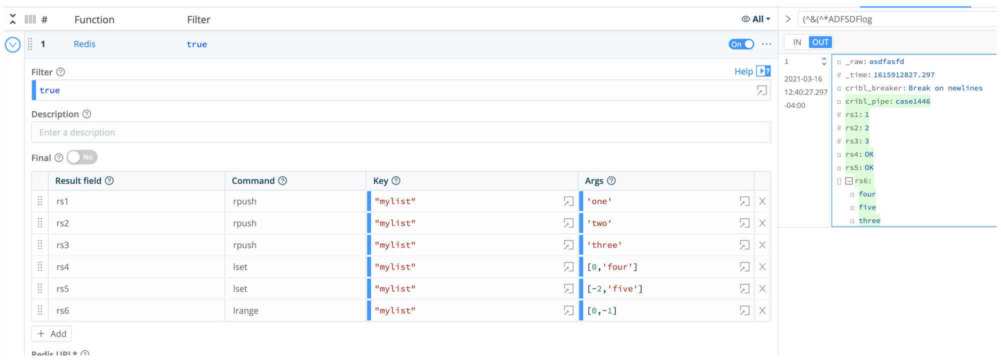
Description: `Read keys from Redis`

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`).



Filter: true

Description: Enter a description

Final: No

Result field	Command	Key	Args
rs1	rpush	"mylist"	'one'
rs2	rpush	"mylist"	'two'
rs3	rpush	"mylist"	'three'
rs4	lset	"mylist"	[0,'four']
rs5	lset	"mylist"	[-2,'five']
rs6	lrange	"mylist"	[0,-1]

```
(^&(^*ADFSDFlog
1
2021-03-16
12:40:27.297
-04:00
_raw: asdfasfd
_time: 1615912827.297
cribl_breaker: Break on newlines
cribl_pipe: case1446
rs1: 1
rs2: 2
rs3: 3
rs4: OK
rs5: OK
rs6:
  four
  five
  three
```

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 Stream 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"
      }
    ]
  }
}
```

14.24. Regex Extract

The Regex Extract Function extracts fields using named capture groups. (In Splunk, these will be index-time fields). Fields that start with `__` (double underscore) are special in Cribl Stream. 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

To test Regex Extract Functions use Data Preview to see the events as they flow into and out of the Pipeline.

Example 1: Single Field from Simple Event

Assume a simple event that looks like this: `metric1=23 metric2=42 dc=23 abc=xyz`

Extract **only** the `metric1` field:

Regex: `metric1=(?<metric1>\d+)` **Result**: `metric1:"23"`

Example 2: Extracting Multiple Fields

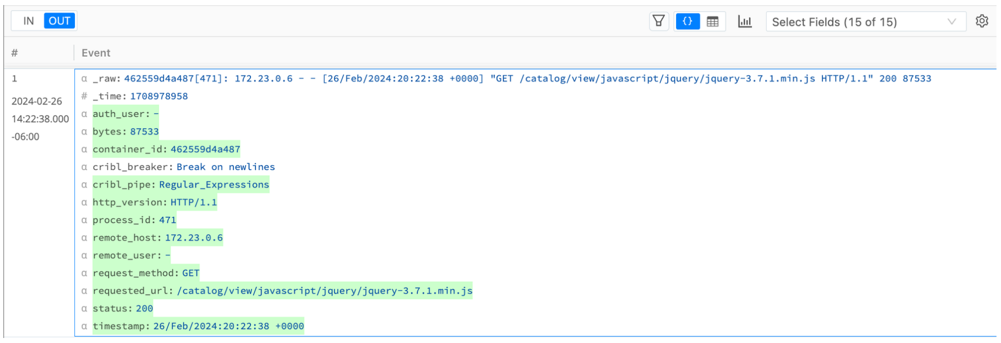
Use this sample data:

```
462559d4a487[471]: 172.23.0.6 - - [26/Feb/2024:20:22:38 +0000] "GET /catalog/view/j
```

Use a regex to extract the fields you know are in the event.

Regex: `^(?<container_id>[^\s]+)\[(?<process_id>\d+)\]:\s+(?<remote_host>[^\s]+)\s+(?<remote_user>-)\s+(?<auth_user>-)\s+\[(?<timestamp>[^\]]+)\]\s+"(?<request_method>\w+)\s+(?<requested_url>[^\s]+)\s+(?<http_version>[^"]+)"\s+(?<status>\d+)\s+(?<bytes>.+)$`

Results:



```
1                      α _raw: 462559d4a487[471]: 172.23.0.6 - - [26/Feb/2024:20:22:38 +0000] "GET /catalog/view/javascript/jquery/jquery-3.7.1.min.js HTTP/1.1" 200 87533
2024-02-26             # _time: 1708978958
14:22:38.000           α auth_user: -
-06:00                 α bytes: 87533
                       α container_id: 462559d4a487
                       α cribl_breaker: Break on newlines
                       α cribl_pipe: Regular_Expressions
                       α http_version: HTTP/1.1
                       α process_id: 471
                       α remote_host: 172.23.0.6
                       α remote_user: -
                       α request_method: GET
                       α requested_url: /catalog/view/javascript/jquery/jquery-3.7.1.min.js
                       α status: 200
                       α timestamp: 26/Feb/2024:20:22:38 +0000
```

Example 2 results

Example 3: Key-Value Pairs from Multiple Fields

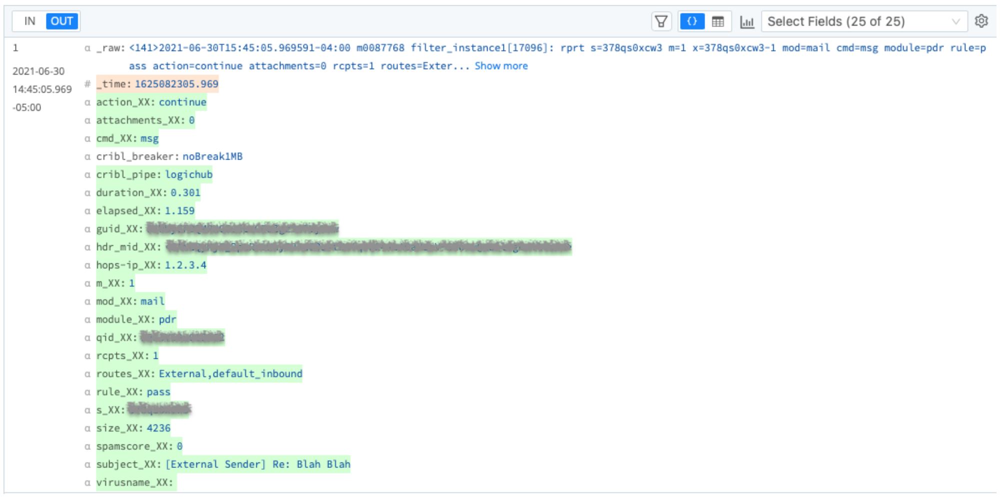
Use this sample data:

```
rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src="00000000-000(
```

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:



```
1                       _raw: <141>2021-06-30T15:45:05.969591-04:00 m0087768 filter_instance1[17096]: rprt s=378qs0xcw3 m=1 x=378qs0xcw3-1 mod=mail cmd=msg module=pdr rule=p
2021-06-30                    ass action=continue attachments=0 rcpts=1 routes=Exter... Show more
14:45:05.969            _time: 1625082305.969
-05:00                  action_XX: continue
                        attachments_XX: 0
                        cmd_XX: msg
                        cribl_breaker: noBreak1MB
                        cribl_pipe: logichub
                        duration_XX: 0.301
                        elapsed_XX: 1.159
                        guid_XX: [redacted]
                        hdr_mid_XX: [redacted]
                        hops-ip_XX: 1.2.3.4
                        m_XX: 1
                        mod_XX: mail
                        module_XX: pdr
                        qid_XX: [redacted]
                        rcpts_XX: 1
                        routes_XX: External,default_inbound
                        rule_XX: pass
                        s_XX: [redacted]
                        size_XX: 4236
                        spamscore_XX: 0
                        subject_XX: [External Sender] Re: Blah Blah
                        virusname_XX:
```

Example 3 results

Example 4: Multi-Stage Extraction, Complex Events

This example builds on the syntax in Example 3, to tackle a more complex event structure.

In the right **Sample Data** pane, click **Paste** and insert the following sample:

Sample Data

```
<134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept"; conn_
```

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.

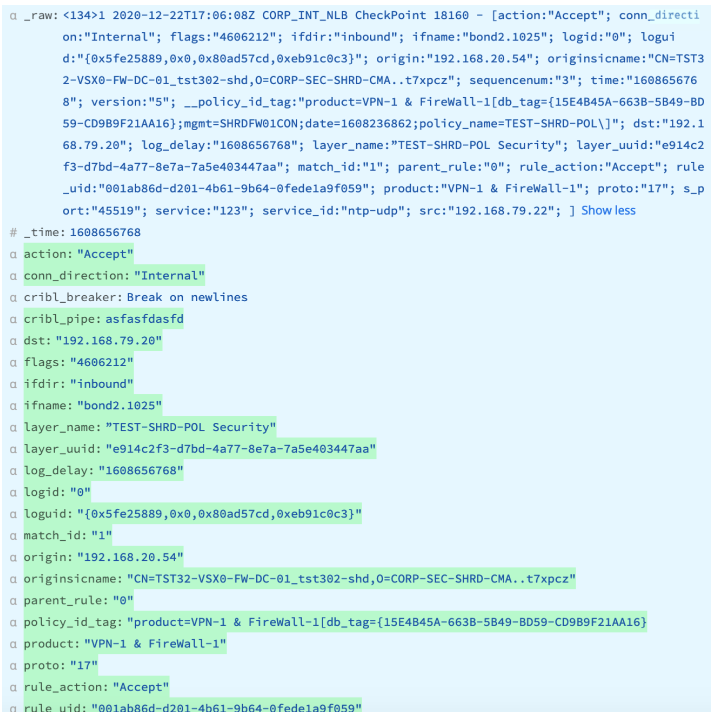
The first Regex Function splits the event to separate the actual data from the header information. We'll split after the `CheckPoint 18160` string, by capturing everything between the `[` and `]`:

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 4 results

For further examples, see Using Cribl to Analyze DNS Logs in Real Time – Part 2.

14.25. 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.

14.26. 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')`

14.27. 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 Stream'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`.

14.28. 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.

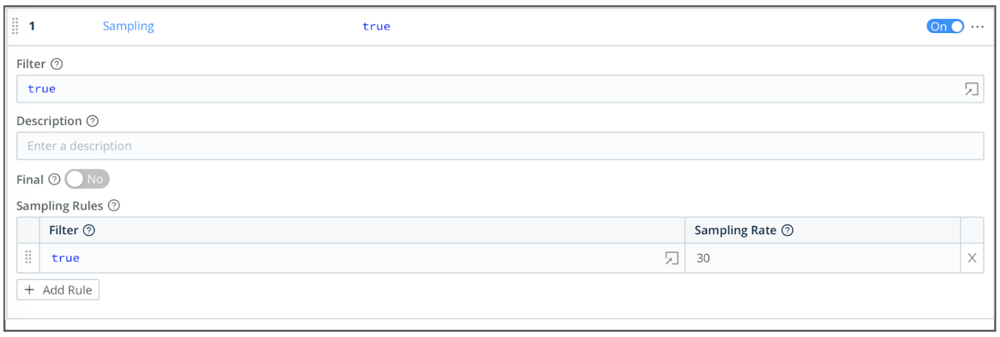
Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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.

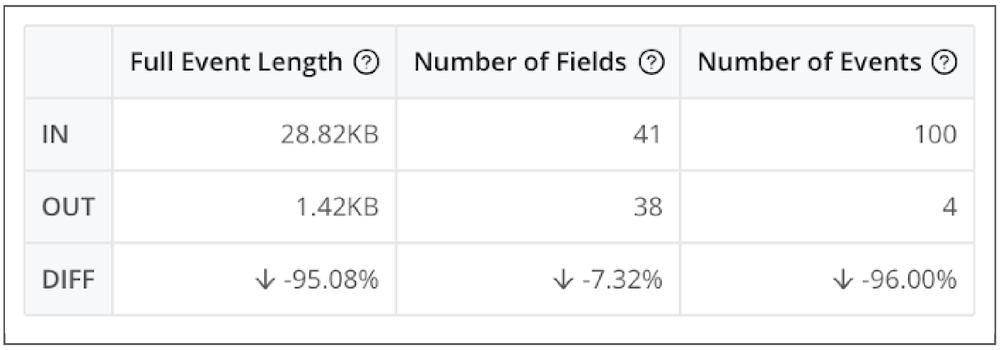
How It Works

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.

14.29. Serialize

Use the Serialize Function to serialize an event's content into a predefined format.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Type: Data output format. Defaults to `CSV`. Options:

- **CSV**
- **Extended Log File Format** – Extended Log File Format.
- **Common Log Format** – Common Log Format.
- **Key=Value Pairs** – selecting this type displays additional options:
  - **Clean Fields**: Toggle to `Yes` to clean field names by replacing non-alphanumeric characters with `_`. This will also strip leading and trailing `"` symbols.
  - **Pair delimiter**: Delimiter used to separate `key=value` pairs. Defaults to a single space character. Multiple characters are allowed. Cannot be an equal sign (`=`).
- **JSON Object**
- **Delimited values** – selecting this type displays additional options:
  - **Delimiter**: Delimiter character to split value. If left blank, defaults to comma (`,`). You can also specify pipe (`|`) or tab characters.
  - **Quote char**: Character used to quote literal values. If left blank, defaults to `"`.
  - **Escape char**: Character used to escape delimiter or quote characters. If left blank, defaults to **Quote char**.
  - **Null value**: Field value representing the null value. These fields will be omitted. Defaults to: `–`

Library: Browse Parser/Formatter library.

Fields to serialize: You must specify individual fields with the `CSV`, `ELFF`, `CLF`, and `Delimited values` Types. All other formats support wildcard field lists. With these formats, the default entry here will specify some excluded fields (indicated by `!`), followed by a `*` wildcard to serialize all remaining fields.

All field names or wildcards that you want to exclude from serialization must precede all field names or wildcards that you want to include.

Source field: Field containing the object to serialize. Leave blank to serialize top-level event fields.

Destination field: Field to serialize the data into. Defaults to `_raw`.

Examples

Scenario A: JSON to CSV

Assume a simple event that looks like this: `{"time":"2019-08-25T14:19:10.240Z","channel":"input","level":"info","message":"initializing input","type":"kafka"}`

We want to serialize these fields: `_time`, `channel`, `level`, and `type` into a single string, in CSV format, stored in a new destination field called `test`.

To properly extract the key-value pairs from this event structure, we'll use a built-in Event Breaker:

1. Copy the above sample event to your clipboard.
2. In the **Preview** pane, select **Paste a Sample**, and paste in the sample event.
3. Under **Select Event Breaker**, choose **ndjson** (newline-delimited JSON), and click **Save as a Sample File**.

Now you're ready to configure the Serialize Function, using the settings below:

Type: `CSV` Fields to Serialize: `_time channel level type` Destination Field: `test` Source Field: [leave empty] **Result**: `test: 1566742750.24,input,info,kafka`

In the new `test` field, you now see the `time`, `channel`, `level`, and `type` keys extracted as top-level fields.

Scenario B: CSV to JSON

Let's assume that a merchant wants to extract a subset of each customer order, to aggregate anonymized order statistics across their customer base. The transaction data is originally in CSV format, but the statistical data must be in JSON.

Here's a CSV header (which we don't want to process), followed by a row that represents one order:

`orderID,custName,street,city,state,zip 20200622102822,john smith,100 Main St.,Anytown,AK,99911`

To convert to JSON, we'll need to first parse each field from the CSV to a manipulable field in the Pipeline, which the Serialize Function will be able to reference. In this example, the new manipulable field is `message`.

Use the `Parser` Function:

Filter: `true` Operation mode: `Extract` Type: `CSV` Source field: `_raw` Destination field: `message` List of fields: `orderID custName street city state zip`

Now use the Serialize Function:

Filter: `true` Type: `JSON` Fields to serialize: `city state` Source field: `message` Destination field: `orderStats`

14.30. 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

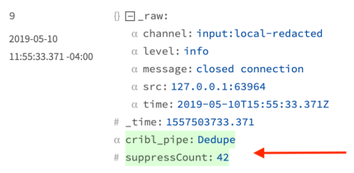
Maximum cache size: The maximum number of keys that can be cached before idle entries are removed. Before changing the default `50000`, contact Cribl Support to understand the implications.

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**.



```
9                        {} ⊟ _raw:
2019-05-10                    α channel: input:local-redacted
11:55:33.371 -04:00           α level: info
                              α message: closed connection
                              α src: 127.0.0.1:63964
                              α time: 2019-05-10T15:55:33.371Z
                         # _time: 1557503733.371
                         α cribl_pipe: Dedupe
                         # suppressCount: 42
```

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.

14.31. 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 Stream worker. This can be useful in verifying the data being processed in a Pipeline.

The Filesystem/NFS Destination offers similar capability, but only after the data leaves the Pipeline. Tee, by comparison, can be inserted at any point in the Pipeline.

In Cribl.Cloud, the Tee Function is only available on hybrid, customer-managed Worker Nodes.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Command: Command to execute and receive events (via `stdin`) – one JSON-formatted event per line.

Args: Click **Add Arg** to supply arguments to the command.

Restart on exit: Restart the process if it exits and/or we fail to write to it. Defaults to `Yes`.

Environment variables: Environment variables to set or overwrite. Click **Add Variable** to add key-value pairs.

Communication Protocol

Data is passed to the command through its `stdin`, using the following protocol:

- First line: Metadata serialized in JSON, containing the following fields:
  - **format**: Serialization format for event. Defaults to `JSON`.
  - **conf**: Full Function configuration.
- Remaining: Payload.

Examples

Assume that we are parsing PANOS Traffic logs, and want to see how they look at a particular step in the processing Pipeline We'll assume that the `Parser` Function is already in place, so we'll insert the Tee Function at any (arbitrary) later point in the Pipeline.

Scenario A:

The Tee Function itself requires only that we define the **Command** field. In this particular example, that **Command** will be `tee` itself.

We've also clicked **Add Arg**, to specify a local output file in the resulting **Args** field. (A file path would normally be the first argument to a `tee` command executed from the command line. The Cribl Stream 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
```

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.

14.32. 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 Stream 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

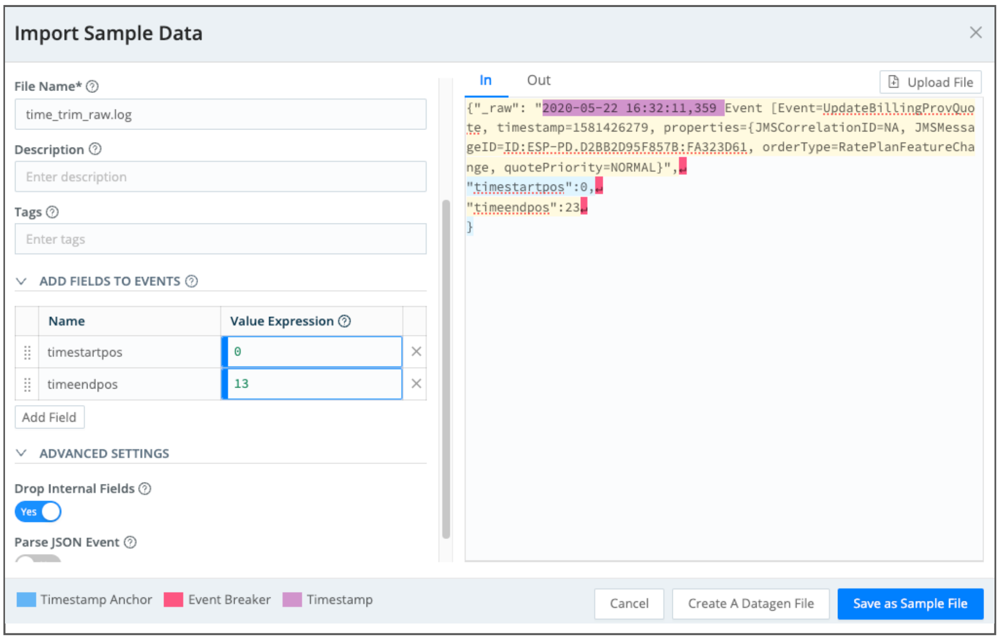
Remove the timestamp pattern (indicated by `timestartpos` and `timeendpos`) from `_raw`, and stash it in a field called `time_field`.

Field name: `time_field`

Example event before:

```
{"_raw": "2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote, timestamp=1
"timestartpos":0,
"timeendpos":23
}
```

To create this example payload, we selected **Sample Data** > **Import**, pasted the `_raw` field's original contents into the resulting modal, and then added the two required position fields:


Import Sample Data

File Name*: time_trim_raw.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

Name	Value Expression
timestartpos	0
timeendpos	13

Add Field

ADVANCED SETTINGS

Drop Internal Fields: Yes

Parse JSON Event

In | Out | Upload File

```
{"_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
}
```

🟦 Timestamp Anchor 🟥 Event Breaker 🟪 Timestamp

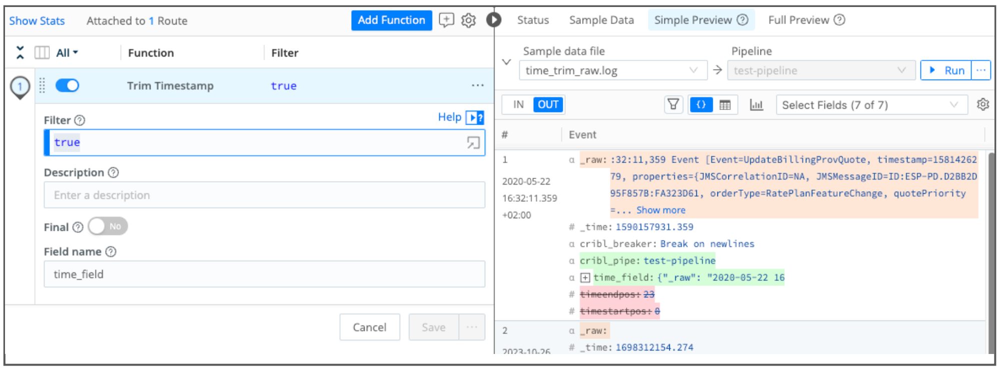
Cancel | Create A Datagen File | Save as Sample File

Example event setup

Example event after:

```
{"_raw": "Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties={JM
"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.


Show Stats | Attached to 1 Route | Add Function

Function	Filter
Trim Timestamp	true

Filter: true

Description: Enter a description

Final: No

Field name: time_field

Cancel | Save

Status | Sample Data | Simple Preview | Full Preview

Sample data file: time_trim_raw.log → Pipeline: test-pipeline | Run

IN | OUT | Select Fields (7 of 7)

```
#  Event
1  _raw: :32:11,359 Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.D2BB2D95F857B:FA323D61, orderType=RatePlanFeatureChange, quotePriority=... Show more
2020-05-22 16:32:11.359 +02:00
   _time: 1590157931.359
   cribl_breaker: Break on newlines
   cribl_pipe: test-pipeline
   time_field: {"_raw": "2020-05-22 16
   ~~timeendpos: 23~~
   ~~timestartpos: 0~~
2  _raw:
2023-10-26
   _time: 1698312154.274
```

Example event, saved and transformed

14.33. Unroll

The Unroll Function accepts an array field – or an expression to evaluate an array field – and breaks/unrolls the array into individual events.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Source field expression: Field in which to find/calculate the array to unroll. E.g.: `_raw`, `_raw.split(/\n/)`. Defaults to `_raw`.

Destination field: Field (within the destination event) in which to place the unrolled value. Defaults to `_raw`.

Example

Assume we want to break/unroll each line of this event:

Sample Event

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02 /lib/systemd/syste
root         2  0.0  0.0      0     0 ?        S     2018   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S     2018   1:51 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<    2018   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S     2018   3:55 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S     2018   0:00 [rcu_bh]
```

Settings

Source field expression: `_raw.split(/\n/)`

The `split()` JavaScript method breaks `_raw` into an ordered set of substrings/values, puts these values into an array, and returns the array.

Destination field: `_raw`

Resulting Events

```
Event 1:
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND

Event 2:
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02 /lib/systemd/syste

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]
```

14.34. 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 Stream 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
<?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 Stream 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>
```

14.35. Prometheus Publisher (Deprecated)

> This Function was deprecated as of Cribl Stream 3.0, and has been removed from Cribl Stream 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 Stream, or metrics derived from the output of Cribl Stream'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 Stream version, the endpoint is: `http://<worker_node_IP>:<api-port>/metrics`. Within Cribl Stream, that endpoint redirects from `http://<worker_node_IP>:9000/metrics` to `http://<worker_node_IP>:9000/api/v1/metrics`.

> If used, this Function **must** follow any Publish Metrics or Aggregations Functions within the same Pipeline. This is to ensure that any data **not** originating from a metrics input is transformed into metrics format.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Fields to publish: Wildcard list of fields to publish to the Prometheus endpoint.

Advanced Settings

Batch write interval: How often, in milliseconds, the contents should be published. Defaults to `5000`.

Passthrough mode: If set to `No` (the default), overrides the **Final** setting, and suppresses output to downstream Functions' Destinations. Toggle to `Yes` to allow events to flow to consumers beyond the Prometheus endpoint. In effect, when previewing the pipeline output what you'll see is your event fields will have strikethrough font applied to them. This does not mean the Prometheus function is not matching your events but rather indicative of the Passthrough being disabled.

Update mode: On the default `No` setting, suppresses output to downstream Functions' Destinations. (This overrides the **Final** setting.) Toggle to `Yes` to allow events to flow to consumers beyond the Prometheus endpoint.

Example

This example uses the same PANOS sample data as the Publish Metrics Function, and is similarly preceded in a Pipeline by a Parser Function that extracts fields from the PANOS log.

Filter: Set as appropriate. **Fields to publish**: Set as appropriate. We'll use the default of `*` for this example. **Advanced settings**: Accept defaults.

After committing and deploying changes, you should be able to use a `curl` command (-L needed to follow the redirect mentioned above) to verify that metrics are being published, just a few seconds after data is ingested on an idle system.

curl output

```
$ curl -L http://<worker_node_IP>:9000/metrics
# TYPE perf_192_168_1_248_bytes_sent counter
metric_192_168_1_248_bytes_sent {destination_ip="160.177.222.249",inbound_interface

# TYPE perf_192_168_1_248_bytes_rcvd counter
metric_192_168_1_248_bytes_rcvd {destination_ip="160.177.222.249",inbound_interface

# TYPE perf_192_168_1_248_pkts_sent counter
metric_192_168_1_248_pkts_sent {destination_ip="160.177.222.249",inbound_interface=

# TYPE perf_192_168_1_248_pkts_rcvd counter
metric_192_168_1_248_pkts_rcvd {destination_ip="160.177.222.249",inbound_interface=
```

Now, we need to have Prometheus scrape the metrics. In this very basic example, you can add the target endpoint to the `prometheus.yml` file, under the `scrape_configs` -> `static_configs` section. Specify the endpoint in `IP:port` syntax, because Prometheus assumes (and requires) `/metrics` for all endpoints.

Restart Prometheus. Within just a few seconds, you should be able to use its query interface to retrieve metrics published by Cribl Stream.

14.36. 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 Stream 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.

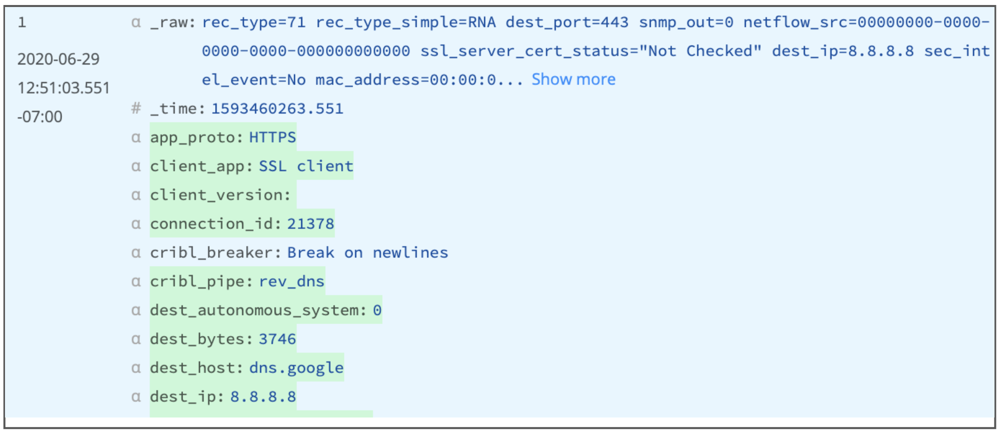
If the field value is not in IPv4 or IPv6 format, the lookup is skipped.

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

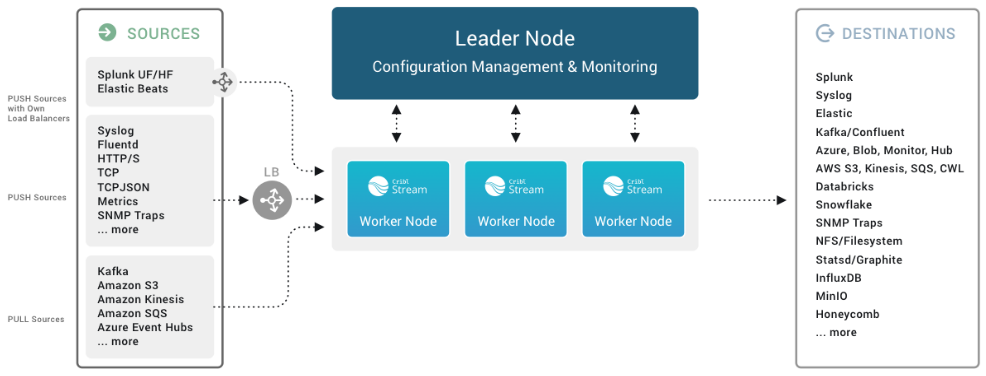
Lookup field name: `dest_ip` **Output field name**: `dest_host` **Result**: See the `dest_ip` field, and the newly created `dest_host` field, in the events.



```
1                α _raw: rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src=00000000-0000-
2020-06-29               0000-0000-000000000000 ssl_server_cert_status="Not Checked" dest_ip=8.8.8.8 sec_int
12:51:03.551             el_event=No mac_address=00:00:0... Show more
-07:00           # _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
```

15. Sources Overview

Cribl Stream can receive continuous data input from various Sources, including Splunk, HTTP, Elastic Beats, Kinesis, Kafka, TCP JSON, and many others. Sources can receive data from either IPv4 or IPv6 addresses.



Push and Pull Sources

Sources Summary

Cribl Stream supports a variety of HTTP-, TCP-, and UDP-based Sources. All HTTP-based Sources are proxyable.

Collector Sources

Collectors ingest data **intermittently** in on-demand bursts (“ad hoc collection”), on preset schedules, or by “replaying” data from local or remote stores.

All Collector Sources are proxyable.

These Sources can ingest data only if a Leader is active:

- Amazon S3 (HTTPS only)
- Azure Blob Storage (HTTPS only)
- Database
- File System/NFS
- Google Cloud Storage (HTTPS only)
- Health Check (HTTP/S)
- REST/API Endpoint (HTTP/S)

- Script (executes any shell script you provide and returns the output of that script/command)
- Splunk Search (HTTP/S)

For background and instructions on using Collectors, see:

- Collectors
- Scheduling and Running
- Job Limits

Check out the example REST Collector configurations in Cribl's Collector Templates repository. For many popular Collectors, the repo provides configurations (with companion Event Breakers, and event samples in some cases) that you can import into your Cribl Stream instance, saving the time you'd have spent building them yourself.

Push Sources

Supported data Sources that **send** to Cribl Stream.

These Sources can continue ingesting data even if no Leader is active:

- Amazon Kinesis Firehose (HTTPS only)
- Datadog Agent (HTTP/S)
- Elasticsearch API (HTTP/S)
- Grafana (HTTP/S)
- HTTP/S (Bulk API)
- Raw HTTP/S
- Loki (HTTP/S)
- Metrics (TCP or UDP)
- Model Driven Telemetry (gPRC)
- NetFlow (NetFlow v5 over UDP)
- OpenTelemetry (OTel) (gRPC or HTTP/S)
- Prometheus Remote Write (HTTP/S)
- SNMP Trap (UDP)
- Splunk HEC (HTTP/S)
- Splunk TCP

- Syslog (TCP or UDP)
- TCP JSON
- TCP (Raw)
- UDP (Raw)
- Windows Event Forwarder (HTTP/S)

Data from these Sources is normally sent to a set of Cribl Stream Workers through a load balancer. Some Sources, such as Splunk forwarders, have native load-balancing capabilities, so you should point these directly at Cribl Stream.

Pull Sources

Supported Sources that Cribl Stream **fetches** data from.

These Sources can ingest data only if a Leader is active:

- Amazon Kinesis Streams (HTTPS only)
- Amazon SQS (HTTPS only)
- Amazon S3 (HTTPS only)
- Google Cloud Pub/Sub (HTTPS only)
- Azure Event Hubs (TCP)
- Azure Blob Storage (HTTPS only)
- Confluent Cloud (TCP)
- CrowdStrike FDR (HTTPS only)
- Office 365 Services (HTTPS only)
- Office 365 Activity (HTTPS only)
- Office 365 Message Trace (HTTPS only)
- Prometheus Scraper (HTTP/S)
- Kafka (TCP)
- Amazon MSK (TCP)
- Splunk Search (HTTP/S)

System Sources

Sources supply information generated by Cribl Stream about itself or move data among Workers within your Cribl Stream deployment.

- AppScope (TCP or Unix socket)
- Datagen
- Exec (on-prem only)
- File Monitor (on-prem only)
- Journal files (on-prem only)
- Kubernetes Logs
- Kubernetes Metrics
- System Metrics (on-prem only)
- System State (on-prem only)

Internal Sources

Similar to System Sources, Internal Sources also supply information generated by Cribl Stream about itself. However, unlike System Sources, they don't count towards the license usage.

- Cribl HTTP (HTTP/S)
- Cribl TCP
- Cribl Internal

Configuring and Managing Sources

For each Source *type*, you can create multiple definitions, depending on your requirements.

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select the desired Source. Next, click either **Add Destination** or (if displayed) **Select Existing**.

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 the desired Source. Next, click **New Source** to open a **New Source** modal.

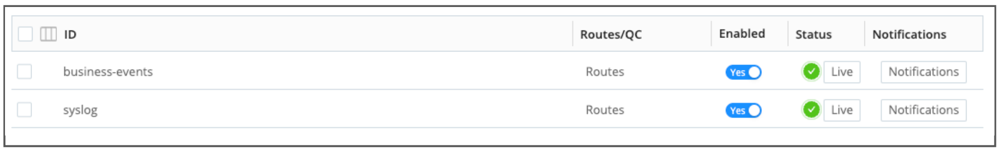
To edit any Source's definition in a JSON text editor, click **Manage as JSON** at the bottom of the **New Source** modal, or on the **Configure** tab when editing an existing Source. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to copy and modify existing Source configurations as `.json` files.

When JSON configuration contains sensitive information, it is redacted during export.

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.

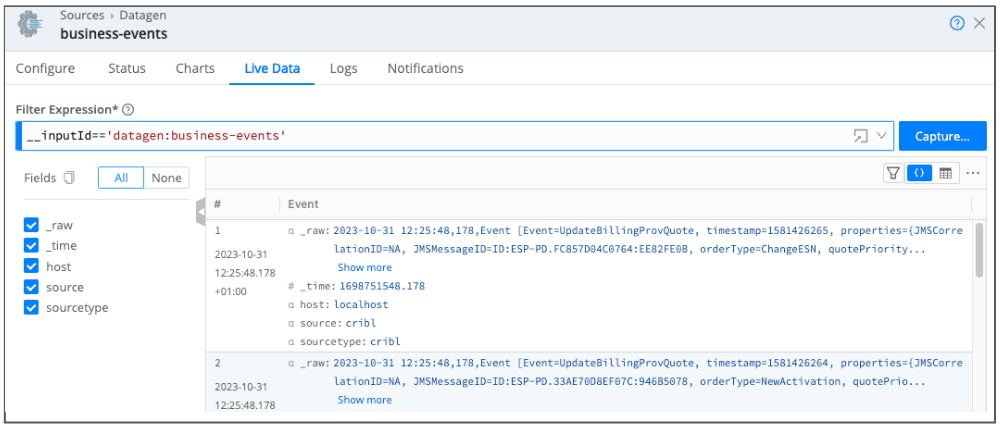
To capture live data, you must have Worker Nodes registered to the Worker Group for which you're viewing events. You can view registered Worker Nodes from the **Status** tab in the Source.



ID	Routes/QC	Enabled	Status	Notifications
business-events	Routes	Yes	✓ Live	Notifications
syslog	Routes	Yes	✓ Live	Notifications

Source > Live button

You can also start an immediate capture from within an enabled Source's config modal, by clicking the modal's **Live Data** tab.



```
__inputId=='datagen:business-events'
```

```
1                 α _raw: 2023-10-31 12:25:48,178,Event [Event=UpdateBillingProvQuote, timestamp=1581426265, properties={JMSCorre
2023-10-31              lationID=NA, JMSMessageID=ID:ESP-PD.FC857D04C0764:EE82FE0B, orderType=ChangeESN, quotePriority...
12:25:48.178            Show more
+01:00            # _time: 1698751548.178
                  α host: localhost
                  α source: cribl
                  α sourcetype: cribl
2                 α _raw: 2023-10-31 12:25:48,178,Event [Event=UpdateBillingProvQuote, timestamp=1581426264, properties={JMSCorre
2023-10-31              lationID=NA, JMSMessageID=ID:ESP-PD.33AE70D8EF07C:946B5078, orderType=NewActivation, quotePrio...
12:25:48.178            Show more
```

Source modal > Live Data tab

Monitoring Source Status

Each Source's configuration modal offers two tabs for monitoring: **Status** and **Charts**.

Status Tab

The **Status** tab provides details about the Workers in the group and their status. An icon shows whether the Worker is operating normally.

You can click each Worker's row to see specific information, for example, to identify issues when the Source displays an error. The specific set of information provided depends on the Source type. The data represents only process 0 for each Worker Node.

> The content of the **Status** tab is loaded live when you open it and only displayed when all the data is ready. With a lot of busy Workers in a group, or Workers located far from the Leader, there may be a delay before you see any information.
>
> The statistics presented are reset when the Worker restarts.

Charts Tab

The **Charts** tab presents a visualization of the recent activity on the Source. The following data is available:

- Events in
- Thruput in (events per second)
- Bytes in
- Thruput in (bytes per second)

> This data (in contrast with the status tab) is read almost instantly and does not reset when restarting a Worker.

Preconfigured Sources

To accelerate your setup, Cribl Stream ships with several common Sources configured for typical listening ports, but not switched on. Open, clone (if desired), modify, and enable any of these preconfigured Sources to get started quickly:

- Syslog – TCP Port 9514, UDP Port 9514
- Splunk TCP – Port 9997
- Splunk HEC – Port 8088
- TCP JSON – Port 10070
- TCP – Port 10060
- HTTP – Port 10080
- Elasticsearch API – Port 9200
- SNMP Trap – Port 9162 (preconfigured only on on-prem setups)
- OpenTelemetry - Port 4317 (preconfigured only on Cribl.Cloud)

System and Internal Sources:

- Cribl Internal > CriblLogs (preconfigured only on on-prem setups)
- Cribl Internal > CriblMetrics
- Appscope
- Cribl HTTP – Port 10200 (preconfigured only on distributed setups)
- Cribl TCP – Port 10300 (preconfigured only on distributed setups)
- System Metrics (preconfigured only on on-prem setups)
- System State (preconfigured only on on-prem setups)
- Journal files (preconfigured only on on-prem setups)

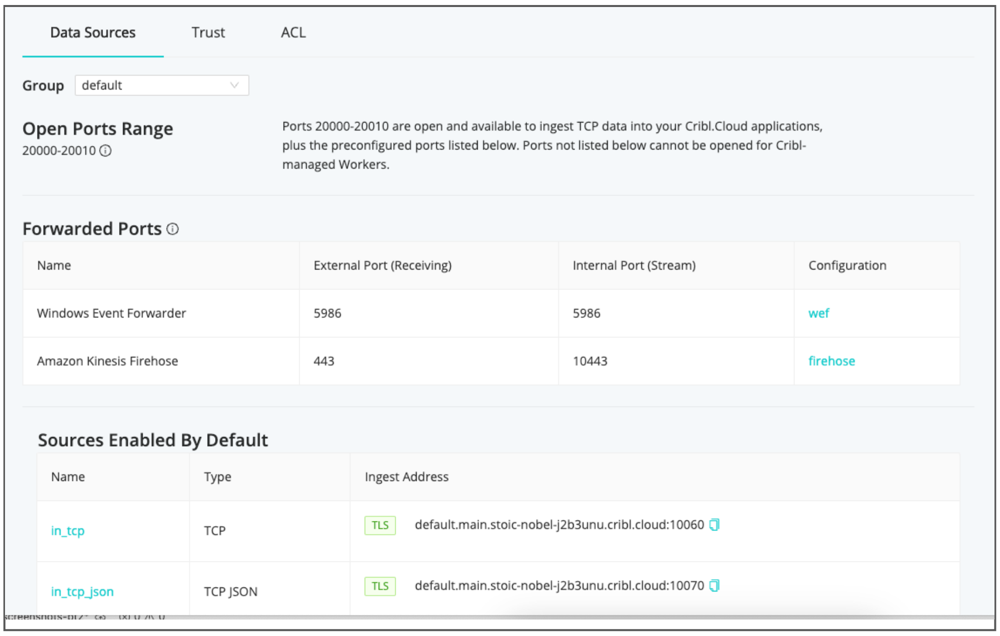
In a preconfigured Source's configuration, never change the **Address** field, even though the UI shows an editable field. If you change these fields' value, the Source will not work as expected.

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 Stream will open the port, and will be able to receive data, immediately.

Cribl.Cloud Ports and TLS Configurations

Cribl.Cloud 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 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 Stream.


Data Sources | Trust | ACL

Group: default

Open Ports Range
20000-20010

Ports 20000-20010 are open and available to ingest TCP data into your Cribl.Cloud applications, plus the preconfigured ports listed below. Ports not listed below cannot be opened for Cribl-managed Workers.

Forwarded Ports

Name	External Port (Receiving)	Internal Port (Stream)	Configuration
Windows Event Forwarder	5986	5986	wef
Amazon Kinesis Firehose	443	10443	firehose

Sources Enabled By Default

Name	Type	Ingest Address
in_tcp	TCP	TLS default.main.stoic-nobel-j2b3unu.cribl.cloud:10060
in_tcp_json	TCP JSON	TLS default.main.stoic-nobel-j2b3unu.cribl.cloud:10070

Available ports and TLS certificates

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 Worker 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.)

Backpressure Behavior and Persistent Queues

By default, a Cribl Stream Source will respond to a **backpressure** situation by blocking incoming data. Backpresssure triggers exist when an in-memory buffer is full and/or when downstream Destinations/receivers are unavailable. The Source will refuse to accept new data until it can flush its buffer.

This will propagate block signals back to the sender, if it supports backpressure. Note that UDP senders (including SNMP Traps and some syslog senders) do not provide this support. In this situation, Cribl Stream will simply drop new events until the Source can process them.

Persistent Queues

Push Sources' config modals provide a **Persistent Queue Settings** option to minimize loss of inbound streaming data. Here, the Source will write data to disk until its in-memory buffer recovers. Then, it will drain the disk-queued data in FIFO (first in, first out) order.

When you enable Source PQ, you can choose between two trigger conditions: **Smart** Mode will engage PQ upon backpressure from Destinations, whereas **Always On** Mode will use PQ as a buffer for **all** events.

For details about the PQ option and these modes, see Persistent Queues.

Persistent queues, when engaged, slow down data throughput somewhat. It is redundant to enable PQ on a Source whose upstream sender is configured to safeguard events in its own disk buffer.

Other Backpressure Options

The S3 Source provides a configurable **Advanced Settings** > **Socket timeout** option, to prevent data loss (partial downloading of logs) during backpressure delays.

Diagnosing Backpressure Errors

When backpressure affects HTTP Sources (Splunk HEC, HTTP/S, Raw HTTP/S, and Kinesis Firehose), Cribl Stream internal logs will show a `503` error code.

15.1. Collector Sources

Unlike other Cribl Stream Sources, Collectors are designed to ingest data intermittently, rather than continuously. You can use Collectors to dispatch on-demand (ad hoc) collection tasks, which fetch or “replay” (re-ingest) data from local or remote locations.

Collectors also support **scheduled** periodic collection jobs – recurring tasks that can make batch collection of stored data more like continual processing of streaming data. You configure Collectors prior to, and independently from, your configuration of ad hoc versus scheduled collection runs.

Collectors are integral to Cribl Stream’s larger story about optimizing your data throughput. Send full-fidelity log and metrics data (“everything”) to low-cost storage, and then use Cribl Stream Collectors to selectively route (“replay”) only needed data to your systems of analysis.

Collector Resources

- Video introduction to Data Collection, in < 2 minutes.
- Video introduction to Data Collection Scheduling, in < 2 minutes.
- Free, interactive try-out of Collectors in Cribl’s Data Collection & Replay sandbox.
- Example Collector configurations – ready to import into Cribl Stream – in Cribl’s Collector Templates repository.
- Using Collectors guides: S3 Storage and Replay | REST API Collectors | Microsoft Graph API Collection | ServiceNow API Collection | Creating a Custom Collector.

Collector Types

Cribl Stream currently provides the following Collector options:

- Azure Blob – enables data collection and replay from Azure Blob Storage objects.
- Cribl Lake - enabled data collection and reply from Cribl Lake datasets.
- Database – enables data collection from database management systems like MySQL and SQL Server.
- File System/NFS – enables data collection and replay from local or remote filesystem locations.
- Google Cloud Storage – enables data collection and replay from Google Cloud Storage buckets.
- Health Check – monitors the availability of system endpoints.
- REST/API Endpoint – enables data collection and replay via REST API calls. Provides four Discover options, to support progressively more complex (and dynamic) item enumerations.

- S3 – enables data collection and replay from Amazon S3 buckets or S3-compatible stores.
- Script – enables data collection and replay via custom scripts.
- Splunk Search – enables data collection and replay from Splunk queries. Supports both simple and complex queries, as well as real-time searches.

If you are exploring Collectors for the first time, the File System/NFS Collector is the simplest to configure, while the REST/API Collector offers the most complex configuration options.

How Do Collectors Work

You can configure a Cribl Stream Node to retrieve data from a remote system by selecting **Manage** from the top nav, then a **Worker Group** to configure. Next, click **Data** > **Sources** > **Collectors**. Data collection is a multi-step process:

First, define a Collector instance. In this step, you configure **collector-specific settings** by selecting a Collector type and pointing it at a specific target. (E.g., the target will be a directory if the type is File System, or an S3 bucket/path if the type is Amazon S3.)

Next, schedule or manually run the Collector. In this step, you configure either scheduled-job–specific or run-specific settings – such as the run Mode (Preview, Discovery, or Full Run), the Filter expression to match the data against, the time range, etc.

When a Node receives this configuration, it prepares the infrastructure to execute a collection job.
A collection job is typically made up of one or more tasks that: discover the data to be fetched; fetch data that match the run filter; and finally, pass the results either through the Routes or (optionally) into a specific Pipeline and Destination.

Select **Monitoring** > **System** > **Job Inspector** to see the results of recent collection runs. You can filter the display by Worker Group (in distributed deployments), and by run type and run timing.

Advanced Collector Configuration

You can edit the configuration of an existing Collector. Or, you can create a new Collector from a template. A template is just a Collector configuration file (in JSON, as usual) intended to copied and edited.

Editing an Existing Collector

1. When configuring the Collector, click **Manage as JSON** on the **Configure** tab.
2. Cribl Stream will open a JSON editor.

3. Edit the Collector as desired.
4. (Optional) If you want to make the Collector configuration available locally, click **Export**.

   When JSON configuration contains sensitive information, it is redacted during export.

5. Click **OK** to exit the **Manage as JSON** modal.
6. Finish configuring the Collector and click **Save**.

Creating a New Collector from a Template

You can create a new Collector from a template like those in Cribl's Collector Templates repository.

For many popular Collectors, the Collector Templates repository provides configurations (with companion Event Breakers, and event samples in some cases) that you can import into your Cribl Stream instance, saving the time you'd have spent building them yourself. For many Collectors, you will need to import both the Collector itself (a `collector.json` file) and its Event Breaker (a `breaker.json` file). See the Event Breakers topic for instructions on how to import them.

Collector configurations can contain placeholders defined in the form `<Label|Description>`, where `Label` is the input field label and `Description` is the tooltip text. The Cribl Stream JSON editor makes it easy to work with placeholders, as you'll see in the following procedure and placeholders example.

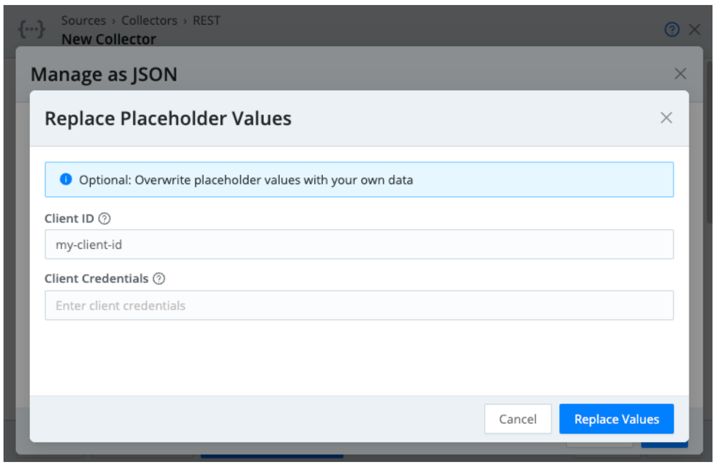
1. Click **Manage as JSON** at the bottom of the **New Collector** modal.
2. Cribl Stream will open a JSON editor.
3. Import the Collector configuration (that is, the template) using either of the two following methods:
   - **Import method**: If your desired Collector configuration file is available locally, click **Import**, navigate to the file, select it, and click **Open**.
   - **Copy and paste method**: If your desired Collector configuration file is open in a local text editor, or is in a Git repository, copy the file. (In the Cribl's Collector Templates repo, navigate to the `collector.json` file in the desired Collector's folder, then click the copy icon.) Back in the JSON text editor, paste the Collector configuration.
4. If the template contains placeholders:
   - If you used the import method, Cribl Stream will open the **Replace Placeholder Values** modal.
   - If you used the copy and paste method, click **OK** to open the modal.
5. Enter desired values for any fields defined as placeholders.
6. (Optional) Edit the configuration further as desired.
7. (Optional) If you want to make the Collector configuration available locally, click **Export**.

When JSON configuration contains sensitive information, it is redacted during export.

8. Click **OK** to exit the **Manage as JSON** modal.
9. Finish configuring the Collector and click **Save**.

Placeholders Example

Here's an example where the configuration defines placeholders for fields named `Client ID` and `Client Credentials`:



Optional: Overwrite placeholder values with your own data

Client ID: my-client-id

Client Credentials: Enter client credentials

Filling in placeholder information

Filling in fields is optional; if you leave a field empty, its placeholder will remain in the JSON configuration, and you can enter a value later.

Scheduled Collection Jobs

You might process data from inherently non-streaming sources, such as REST endpoints, blob stores, etc. Scheduled jobs enable you to emulate a data stream by scraping data from these sources in batches, on a set interval.

You can schedule a specific job to pick up new data from the source – data that hadn't been picked up in previous invocations of this scheduled job. This essentially transforms a non-streaming data source into a streaming data source.

Collectors in Distributed Deployments

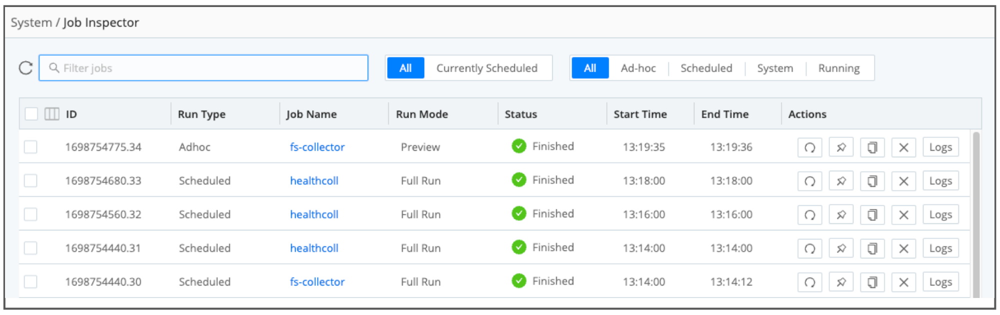
In a distributed deployment, you configure Collectors at the Worker Group level, and Worker Nodes execute the tasks. However, the Leader Node oversees the task distribution, and tries to maintain a fair balance across jobs.

When Workers ask for tasks, the Leader will normally try to assign the next task from a job that has the least tasks in progress. This is known as “Least-In-Flight Scheduling,” and it provides the fairest task distribution for most cases. If desired, you can change this default behavior by opening **Group Settings** > **General Settings** > **Limits** > **Jobs**, and then setting **Job dispatching** to **Round Robin**.

More generally: In a distributed deployment, you configure Collectors and their jobs on individual Worker Groups. But because the Leader manages Collectors’ state, if the Leader instance fails, Collection jobs will fail as well. (This is unlike other Sources, where Worker Groups can continue autonomously receiving incoming data if the Leader goes down.)

Monitoring and Inspecting Collection Jobs

Select **Monitoring** > **System** > **Job Inspector** to view and manage pending, in-flight, and completed collection jobs and their tasks.



ID	Run Type	Job Name	Run Mode	Status	Start Time	End Time	Actions
1698754775.34	Adhoc	fs-collector	Preview	Finished	13:19:35	13:19:36	Logs
1698754680.33	Scheduled	healthcoll	Full Run	Finished	13:18:00	13:18:00	Logs
1698754560.32	Scheduled	healthcoll	Full Run	Finished	13:16:00	13:16:00	Logs
1698754440.31	Scheduled	healthcoll	Full Run	Finished	13:14:00	13:14:00	Logs
1698754440.30	Scheduled	fs-collector	Full Run	Finished	13:14:00	13:14:12	Logs

Job Inspector

Here are the options available on the Job Inspector page:

- **All** vs. **Currently Scheduled** tabs: Click **Currently Scheduled** to see jobs forward-scheduled for future execution – including their cron schedule details, last execution, and next scheduled execution. Click **All** to see all jobs initiated in the past, regardless of completion status.
- Job categories (buttons): Select among **Ad-hoc**, **Scheduled**, **System**, and **Running**. (At this level, **Scheduled** means scheduled jobs already running or finished.)
- Group selectors: Select one or more check boxes to display the **Pause**, **Resume**, etc., buttons shown along the bottom.
- Sortable headers: Click any column to reverse its sort direction.
- Search bar: Click to filter displayed jobs by arbitrary strings.
- Action buttons: For finished jobs, the icons (from left to right) indicate: **Rerun**; **Keep job artifacts**; **Copy job artifacts**; **Delete job artifacts**; and **Display job logs** in a modal. For running jobs, the options

(again from left to right) are: **Pause**; **Stop**; **Copy job artifacts**; **Delete job artifacts**; and **Live** (show collection status in a modal).

15.1.1. Azure Blob Storage

Cribl Stream supports collecting data, and replaying specific events, both from Azure Blob Storage, and from Azure Data Lake Storage Gen2, which implements a hierarchical namespace over blob data. This page covers how to configure the Collector.

Configuring an Azure Blob Storage Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Azure Blob** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **Azure Blob** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Cribl Stream supports data collection and replay from Azure's **hot** and **cool** access tiers, but not from the **archive** tier – whose stated retrieval lag, up to several hours, cannot guarantee data availability.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. For example, `azure_42-a`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Auto-populate from: Optionally, select a predefined Destination that will be used to auto-populate Collector settings. Useful when replaying data.

Container name: Container to collect from. 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}`.

Container names can include only lowercase letters, numbers, and/or hyphens (`-`). This restriction is imposed by Azure.

Authentication

Use the **Authentication method** drop-down 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 Stream 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. 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 to generate a new secret.

Using Shared Access Signature

You can authenticate for Azure Blob Storage using a shared access signature token as connection string.

To employ it successfully, go to your storage account's **Shared access signature** settings and make sure that **Allowed blob index permissions** enables both **Read/Write** and **Filter**.

If you are using Azure Premium storage, you can't set blob index permissions. In such a case, in your collector's setting, enable the **Include Metadata** option. This will allow you to use shared access signature, but as side effect will strip defined tags for downloaded blobs.

Optional Settings

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}/`). More on templates and Filters.

Path extractors: Extractors allow using template tokens as context for expressions that enrich discovery results.

Click **Add Extractor** to add each extractor as a key-value pair, mapping a **Token** name on the left (of the form `/<path>/${<token>}`) to a custom JavaScript **Extractor expression** on the right (e.g., `{host: value.toLowerCase()}`).

Each expression accesses its corresponding `<token>` through the `value` variable, and evaluates the token to populate event fields. Here is a complete example:

Token	Expression	Matched Value	Extracted Result
`/var/log/${foobar}`	`foobar: {program: value.split('.')[0]}`	`/var/log/syslog.1`	`{program: syslog, foobar: syslog.1}`

Recursive: If set to `Yes` (the default), data collection will recurse through subdirectories.

Include metadata: With the default `Yes` setting, Cribl Stream will include Azure Blob metadata in collected events (at `__collectible.metadata`).

Include tags: With the default `Yes` setting, Cribl Stream will include Azure Blob tags in collected events (at `__collectible.tags`). To prevent errors, toggle this to `No` when using a Shared Access Signature connection string, especially on storage accounts that do not support Azure Blob index tags.

Max batch size (objects): Maximum number of metadata objects to batch before recording as results. Defaults to `10`. To override this limit in the Collector's Schedule/Run modal, use Advanced Settings > Upper task bundle size.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

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>`

A fictitious example, using Microsoft's recommended HTTPS option, is:
`DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32`

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained

> and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields: List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Replay

See these resources that demonstrate how to replay data from object storage. Both are written around Amazon S3-compatible stores, but the general principles apply to Azure blobs as well:

- Data Collection & Replay sandbox: Step-by-step tutorial, in a hosted environment, with all inputs and outputs preconfigured for you. Takes about 30 minutes.
- Using S3 Storage and Replay: Guided walk-through on setting up your own replay.

How the Collector Pulls Data

In the Discover phase, the first available Worker returns the list of files to the Leader Node. In the Collect phase, Cribl Stream spreads the list of files to process spread across 1..N Workers, based on file size, with the goal of distributing tasks as evenly as possible across Workers. These Workers then stream in their assigned files from the remote Azure Blob Storage location.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These “meta” fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object’s nested fields contain metadata about each collection job. As mentioned above in Optional Settings, this object will include Azure Blob’s own metadata at `__collectible.metadata` and Azure Blob tags at `__collectible.tags`, unless you deselect the corresponding UI toggles.

Troubleshooting

When permissions are correct on the object store, and events are reaching the Collector, the Preview pane will show events and the Job Inspector will show an **Events collected** count.

However, if previewing returns no events and throws no error, first check your Filter expression by previewing without it (e.g., simplify the Filter expression to `true`). Then check the Job Inspector: If the **Total size** is greater than `0`, and the **Received size** is `NA` or `0`, make sure you have list and read permissions on the object store.

15.1.2. Cribl Lake Collector

The Cribl Lake Collector gathers data from the Cribl Lake.

The Cribl Lake Collector is available only in Cribl.Cloud.

Configure a Cribl Lake Collector

1. From the top nav, click **Manage**, then select a Worker Group to configure.
2. Select **Data**, then **Sources**.
3. In the **Manage Sources** page's tiles or left nav, select **Collectors**, then **Cribl Lake**.
4. Click **Add Collector** to open the **Cribl Lake**, then **New Collector** modal, which provides the following options and fields.
5. The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs.
6. Click **Save** when you've configured your Collector.

You can't use QuickConnect to configure Cribl Lake Collector Sources.

1. In the Source modal, configure the following under **General Settings**:
   - **Collector ID**: Unique ID for this Collector. For example: `myLakeCollector`.
   - **Lake dataset**: Lake dataset to collect data from.
2. Next, you can configure the following **Optional Settings**:
   - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
3. Optionally, configure any Result and Advanced settings outlined in the below sections.
4. Click **Save**, then **Commit & Deploy**.
5. Verify that data is being collected. See the Verify Data Flow section below.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **Throttling**: Rate (in bytes per second) to throttle while writing to an output. Also takes values with multiple-byte units, such as `KB`, `MB`, or `GB`. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained

> and separate from Cribl Stream’s routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job’s artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs’ execution.

Verify Data Flow

To verify that the Collector actually collects data, you can start a single run in the Preview mode.

1. In the **Manage Sources** > **Collectors** > **Cribl Lake** screen, select your Collector’s **Run** action.
2. Make sure mode **Preview** is selected and accept other default settings.
3. Confirm with **Run**.
4. Look at the preview screen to check that data is being collected from Cribl Lake.

15.1.3. Database

You can configure Database Collectors to pull events from database management systems (DBMSs). This enables you to combine such structured data with unstructured machine data, and then route combined data to downstream systems of analysis to gain new insights.

Cribl Stream 4.1 and later support integrating this Collector with MySQL, SQL Server, and Postgres databases.

How the Collector Pulls Data

Like other Cribl Stream Collectors, a Database Collector can perform Preview, Discovery, and Full Run operations in ad hoc or scheduled runs.

Database Collectors rely on Cribl Stream Database Connections Knowledge objects. Before you configure a Collector, configure a Database Connection to negotiate authenticated communication with the appropriate database type.

Database Collectors and Worker Nodes

A Database Collector runs a single `collect` task, which only runs on – and executes a SQL query from – a single Worker Process. However, if the Database Collector is scheduled to run – or run in Preview mode – while the first `collect` task is still running, it will start a simultaneous `collect` task on a different Worker Process.

Configuring a Database Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Database** from the **Manage Sources** page's tiles or left nav. Click **New Collector** to open a modal that provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. For example, `sh2GetStuff`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Connection: Use the drop-down to select a Database Connections already configured on your Cribl Stream installation.

SQL query: Enter a JavaScript expression, enclosed in backticks, that resolves to a query string for selecting data from the database. Supports only a single statement, and only `SELECT`. Has access to the special `${earliest}` and `${latest}` variables, which will resolve to the Collector run's start and end time. Example:

```
`SELECT * FROM table_name WHERE timestamp_field > ${earliest} AND timestamp_field <
```

The Database Collector's `earliest` and `latest` variables are formatted using ISO 8601 conventions, unlike other Collectors' corresponding variables.

You can also craft a query that says, in effect, "give me everything new since my last collection job." To do this, you specify a **tracking column** in the query. The tracking column (also known as a **rising column**) must be one whose values you know always increase from one record to the next. See the explanation and example below.

Optional Settings

Validate query: Enforces strict query validation that allows only a single `SELECT` statement to be entered in the **SQL query** field. Defaults to `Yes`.

Disabling query validation allows potentially destructive queries to be executed, such as data definition language (DDL) and data manipulation language (DML) statements. Such queries can be harmful to your database. Back up your database before running such a query, and proceed with caution.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

Custom Command

In this section, you can pass the data from this Collector to an external command for processing, before the data continues downstream.

Enabled: Defaults to `No`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination to receive the events. The `No` setting exposes these two fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Optionally, use the drop-down to select an existing Pipeline to process results before sending them to Routes.

This final field is always exposed:

- **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`, and so forth. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when you're configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discovery results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Working with a Tracking Column

You can enable state tracking under the `State Tracking` section when scheduling or performing a full run of the Collector. (Preview and Discovery runs do not support state tracking.) With state tracking enabled, the following field will appear:

Tracking Column: A numeric column whose values increase monotonically. The Collector will track the column's greatest value seen in a previous run.

You can configure the Collector to use a **tracking column** (also known as a **rising column**). This prevents overlap between jobs, which occurs when the Collector fetches the same record more than once. It likewise prevents the occurrence of gaps between data collected by consecutive jobs. (If your database already has mechanisms that protect against overlap and gaps, you probably don't need to use a tracking column.)

For your tracking column, you can choose any column whose values always increase from one record to the next. (A sequence of values that behaves like this is said to increase monotonically.) Many database tables have monotonically increasing `id`, `time`, and/or `date` columns.

State tracking is available only for scheduled and full ad-hoc collection (but not for preview or discovery ad-hoc runs). Once you configure your Collector, you can enable state tracking and specify a tracking column while scheduling a job or performing a full ad-hoc run.

To explain the syntax of queries that work with state tracking, we'll break down a few examples.

Tracking Column Queries: A Basic Example

```
`SELECT * FROM my_table WHERE ${state && state.id ? `id > ${state.id.value}` : '1 =
```

The `SELECT` statement must satisfy one requirement: that the tracking column be among the columns named. `SELECT * FROM my_table` satisfies this requirement because the wildcard (`*`) means "all columns."

The `WHERE` clause illustrates the key techniques required for state tracking:

- First, we verify that state has been saved (on a prior run of the Collector); and, if it has, that it includes the desired tracking column – for this example, a column named `id`.

  ```
  WHERE ${state && state.id
  ```

- If state **has** been saved, we'll apply the test right after the `?` in the ternary operator. This determines whether the value of the `id` column for a given row is greater than the last-saved value for that column. If it is not, that means that the row was collected already, and should not be collected again.

  ```
  ? `id > ${state.id.value}` : '1 = 1'}`
  ```

- If state has **not** been saved, this means that the Collector is now about to run **for the first time**.

The example assumes that we want to collect all rows. To do this, we need an expression that always evaluates to true, so every row gets collected. The expression `'1 = 1'` after the ternary operator's colon

(`:`) does exactly that.

Finally, the `ORDER BY` clause, `ORDER BY id`, guarantees that the query returns rows such that the value of the tracking column (here, `id`) increases monotonically from row to row. For state tracking to work correctly, rows **must** be returned such that the value of the tracking column increases monotonically. Some databases have mechanisms that achieve this without an `ORDER BY` clause, but one way or another, this requirement must be satisfied.

Tracking Column Queries: Starting in the Middle

The previous example assumes that you want the Collector's first run to start with the first row of your table. Sometimes, though, you might need the Collector to start somewhere in the middle. If so, you'll need to replace the expression `'1 = 1'` with an expression that finds the desired starting point.

For example:

- If the tracking column is `id` and you want the Collector to begin with the record whose `id` is `4200`, you'd write `id > 4199` instead of `'1 = 1'`.
- Now, the first Collector run will start where `id`'s value is `4200`, and its saved state will include the location where the Collector leaves off.
- In all subsequent runs, the initial test of `state && state.id` will evaluate to true – because there is saved state.
- This means that the Collector will apply `id > ${state.id.value}` and start each new run with next not-yet-collected record.

Tracking Column Queries: Advanced Example

Timestamp columns are a popular choice for state tracking. Here's an example:

```
SELECT id, text_data, timestamp, UNIX_TIMESTAMP(timestamp) as tracking_timestamp FR
WHERE ${state && state.tracking_timestamp ? `UNIX_TIMESTAMP(timestamp) > ${state.tr
ORDER BY tracking_timestamp
```

This query differs from the basic example above in one important way: We use the SQL function `UNIX_TIMESTAMP()` to convert the value in the database's `timestamp` column, from a date to an integer. (This example assumes that we're dealing with a MySQL database, where the `timestamp` column's value will be a date.)

Cribl Stream requires the tracking column's value to be an integer; and here, we satisfy that requirement by using the `UNIX_TIMESTAMP()` function. In your own queries, depending on the nature of the columns you want to use for tracking, you might be able to adapt this syntax – or you might need to get more creative.

Working with Stateful Columns

You can configure a Database Collector to process tables only when new rows have been added. Two configurations are possible – one using stateful sequence columns and one using stateful timestamp columns.

In either case, using a stateful column is a four-step process:

1. Set up a database if you do not already have one.
2. Create a database connection in Cribl Stream.
3. Create a Database Collector.
4. Configure the Database Collector to track the sequence or timestamp.

The first time your Collector runs, it will process all rows in the relevant table. On subsequent runs, it will process only rows added since the last run.

Creating a Database Connection in Cribl Stream

To create a database connection:

1. Select **Processing** > **Knowledge** > **Database Connections**.
2. Click **Add Database Connection**.
3. Give the connection an ID of your choosing.
4. Select the **Database type** appropriate for your application.
5. Enter a connection string that is appropriate for your database. The following example is for a SQL Server database:
   `Server=<server-name>; Database=<database>;User Id=<user>; Password=<password>;trustServerCertificate=true;`
6. Click **Test Connection** to verify the connection.
7. Click **Save**.
8. Commit and deploy your changes.

Stateful Column Queries: Sequence Example

To set up a Database Collector to use stateful sequence columns:

1. Create a Collector.

2. Give the connection an ID of your choosing.
3. Select the database connection you created previously.
4. For **SQL Query**, input the following string, including the backticks:

```
`SELECT * FROM my_table WHERE ${state && state.id ? `id > ${state.id.value}` :
```

1. Click **Save**. The Collector modal will close.
2. Commit and deploy your changes.
3. Reopen your Collector and click **Schedule**.

In the scheduling modal:

1. Enable scheduling for the Collector.
2. Under **State Tracking**, set **Enabled** to `Yes` and **Tracking Column** to the name of the column storing your auto-incrementing ID.
3. Click **Save**.

The Collector will process all rows in the table when you run it for the first time. On subsequent runs, it will process only rows added since the last run.

Stateful Column Queries: Timestamp Example

To set up a Database Collector to use stateful timestamp columns:

1. Create a Collector.
2. Give the connection an ID of your choosing.
3. Select the database connection you created previously.
4. For **SQL Query**, input a string like the following. Include the backticks, and use the table and column names from your own database. In this example, the `DATEDIFF` function is specific to a SQL Server database; the function for your database may differ.
   ```
   `select <id>, <text_data>, <timestamp>, DATEDIFF(s, '1970-01-01 00:00:00', timestamp) as tracking_timestamp from <table-name> where ${state && state.tracking_timestamp ? `DATEDIFF(s, '1970-01-01 00:00:00', timestamp) > ${state.tracking_timestamp.value}` : "1=1"} order by tracking_timestamp`
   ```
5. Click **Save**. The Collector modal will close.
6. Commit and deploy your changes.
7. Reopen your Collector and click **Schedule**.

In the scheduling modal:

1. Enable scheduling for the Collector.
2. Under **State Tracking**, set **Enabled** to `Yes` and **Tracking Column** to the name of the column that contains the timestamp.
3. Click **Save**.

The Collector will process all rows in the table when you run it for the first time. On subsequent runs, it will process only rows added since the last run.

In this example the SQL Server function `DATEDIFF(s, '1970-01-01 00:00:00', <timestampColumn>)` must be used to convert the database date value to a numeric Epic (seconds since Epoch) value. This conversion is needed because most databases return a formatted timestamp string `mm/dd/yyyy hh:mm:ss (2023-07-11 14:24:18)`, which must be converted to a number for state tracking. In this case, Cribl Stream returns the raw timestamp column as well as the timestamp converted to a number, which is required for state tracking.

Use a Function like Eval to remove `tracking_timestamp` from events if you don't want to send this field to a Destination.

Manage Collector State

See the Manage Collector State topic for instructions on editing or deleting Collector state.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These "meta" fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object's nested fields contain metadata about each collection job.

15.1.4. File System/NFS

Cribl Stream supports collecting data from a locally mounted filesystem location that is available on all Worker Nodes.

> In Cribl.Cloud, the File System/NFS Collector is only available on hybrid, customer-managed Worker Nodes.

Configuring a File System Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **File System** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **File System** > **New Collector** modal, which provides the following options and fields.

> The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.
>
> Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. E.g., `DysonV11Roomba960`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID.**

Auto-populate from: Select a Destination with which to auto-populate Collector settings. Useful when replaying data.

Directory: The directory from which to collect data. Templating is supported (e.g., `/myDir/${host}/${year}/${month}/`). You can also use templating to specify (e.g.) a Splunk bucket from which to collect. Symlinks will not be followed. More on templates and Filters.

Optional Settings

Path extractors: Extractors allow using template tokens as context for expressions that enrich discovery results.

Click **Add Extractor** to add each extractor as a key-value pair, mapping a **Token** name on the left (of the form `/<path>/${<token>}`) to a custom JavaScript **Extractor expression** on the right (e.g., `{host: value.toLowerCase()}`).

Each expression accesses its corresponding `<token>` through the `value` variable, and evaluates the token to populate event fields. Here is a complete example:

Token	Expression	Matched Value	Extracted Result
`/var/log/${foobar}`	`foobar: {program: value.split('.')[0]}`	`/var/log/syslog.1`	`{program: syslog, foobar: syslog.1}`

Recursive: If set to `Yes` (the default), data collection will recurse through subdirectories.

Max batch size (files): Maximum number of lines written to the discovery results files each time. Defaults to `10`. To override this limit in the Collector's Schedule/Run modal, use Advanced Settings > Upper task bundle size.

Destructive: If set to `Yes`, the Collector will delete files after collection. Defaults to `No`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Cribl Stream automatically detects gzip compression where a file name ends in `.gz`.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

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.

Advanced Settings enable you to customize post-processing and administrative options.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

How the Collector Pulls Data

When you run a Filesystem/NFS Collector in Discovery mode, the first available Worker returns the list of available files to the Leader Node.

In Full Run mode, the Leader distributes the list of files to process across 1..*N* Workers as evenly as possible, based on file size. These Workers then stream in their assigned files from the filesystem location.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These "meta" fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object's nested fields contain metadata about each collection job.

15.1.5. Google Cloud Storage

Cribl Stream supports collecting data objects from Google Cloud Storage buckets. This page covers how to configure the Collector.

Configuring a Google Cloud Storage Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Google Cloud Storage** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **Google Cloud Storage** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. E.g., `gcs_24-7`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Auto-populate from: Optionally, select a predefined Destination that will be used to auto-populate Collector settings. Useful when replaying data.

Bucket name: Google Cloud Storage bucket to collect from. 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}`.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Manual**: Use this default option to enter your Google Cloud Storage connection string directly. Exposes a **Service account credentials** field for this purpose.
  The **Service account credentials** are the contents of a Google Cloud service account credentials (JSON keys) file. To upload a file, click the upload button at this field's upper right.

- **Secret**: This option exposes a **Service account credentials (text secret)** drop-down, in which you can select a stored secret that references an Google Cloud Storage connection string. 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 to generate a new secret.

You can access service account credentials in the Google Cloud Console under **Service Accounts >** <service account associated with bucket> **> Keys**. The key file must be in JSON format.

Optional Settings

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}/`). More on templates and Filters.

Path extractors: Extractors allow using template tokens as context for expressions that enrich discovery results.

Click **Add Extractor** to add each extractor as a key-value pair, mapping a **Token** name on the left (of the form `/<path>/${<token>}`) to a custom JavaScript **Extractor expression** on the right (e.g., `{host: value.toLowerCase()}`).

Each expression accesses its corresponding `<token>` through the `value` variable, and evaluates the token to populate event fields. Here is a complete example:

Token	Expression	Matched Value	Extracted Result
`/var/log/${foobar}`	`foobar: {program: value.split('.')[0]}`	`/var/log/syslog.1`	`{program: syslog, foobar: syslog.1}`

Endpoint: Google Cloud Storage service endpoint. If empty, the endpoint will be automatically constructed using the service account credentials.

Disable time filter: Toggle to `Yes` if your Run or Schedule configuration specifies a date range and no events are being collected. This will disable the Collector's event time filtering to prevent timestamp conflicts.

Recursive: If set to `Yes` (the default), data collection will recurse through subdirectories.

Max batch size (objects): Maximum number of metadata objects to batch before recording as results. Defaults to `10`. To override this limit in the Collector's Schedule/Run modal, use Advanced Settings > Upper task bundle size.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

> You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Pre-processing Pipeline: Pipeline to process results before sending to Routes. Optional, and available only when **Send to Routes** is toggled to `Yes`.

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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Google Cloud Roles and Permissions

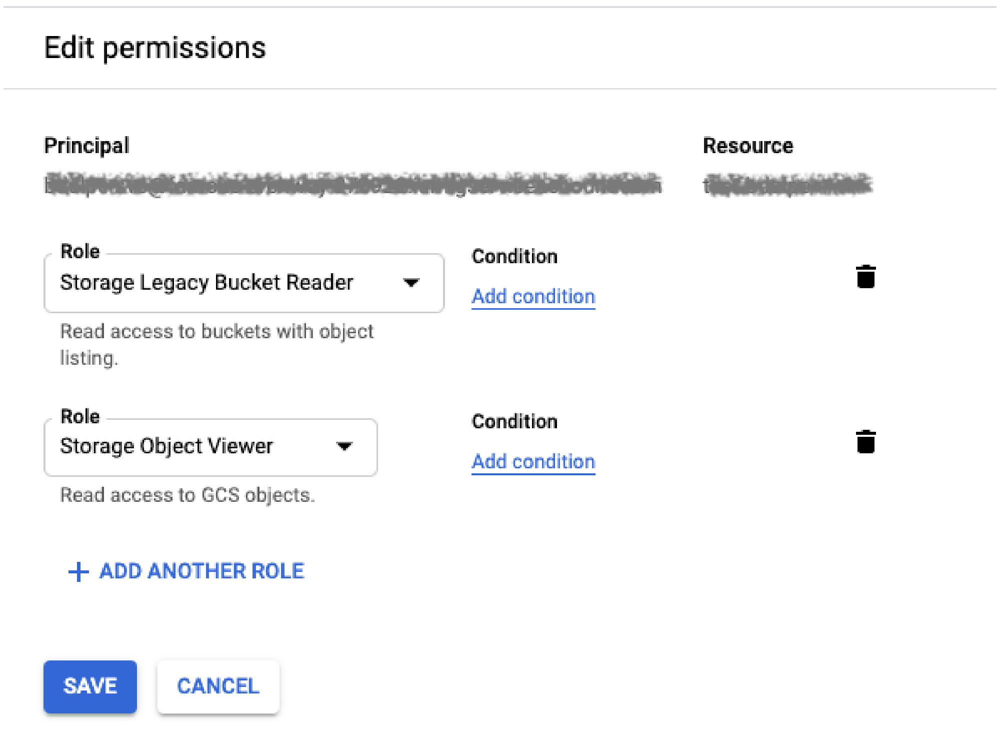
Your Google Cloud service account will need, at a minimum, roles and permissions that enable you to set up a new bucket or modify an existing one:

Roles

- Storage Legacy Bucket Reader: `roles/storage.legacyBucketReader`
- Storage Legacy Object Reader: `roles/storage.legacyObjectReader`

Permissions

- `storage.buckets.get`
- `storage.objects.get`
- `storage.objects.list`



Edit permissions

Principal
[redacted]

Resource
[redacted]

Role
Storage Legacy Bucket Reader
Read access to buckets with object listing.

Condition
Add condition

Role
Storage Object Viewer
Read access to GCS objects.

Condition
Add condition

+ ADD ANOTHER ROLE

SAVE CANCEL

Editing Google Cloud roles and permissions

For additional details, see the Google Cloud Access Control topic.

Replay

See these resources that demonstrate how to replay data from object storage. Both are written around Amazon S3-compatible stores, but the general principles apply to Google Cloud buckets as well:

- Data Collection & Replay sandbox: Step-by-step tutorial, in a hosted environment, with all inputs and outputs preconfigured for you. Takes about 30 minutes.
- Using S3 Storage and Replay: Guided walk-through on setting up your own replay.

How the Collector Pulls Data

In the Discover phase, the first available Worker returns the list of files to the Leader Node. In the Collect phase, Cribl Stream spreads the list of files to process spread across 1..N Workers, based on file size, with the goal of distributing tasks as evenly as possible across Workers. These Workers then stream in their assigned files from the remote Google Cloud Storage location.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These “meta” fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object’s nested fields contain metadata about each collection job.

Troubleshooting

When permissions are correct on the object store, and events are reaching the Collector, the Preview pane will show events and the Job Inspector will show an **Events collected** count.

However, if previewing returns no events and throws no error, first check your Filter expression by previewing without it (e.g., simplify the Filter expression to `true`). Then check the Job Inspector: If the

Total size is greater than `0`, and the **Received size** is `NA` or `0`, make sure you have list and read permissions on the object store.

15.1.6. Health Check

The Health Check Collector monitors endpoints by sending requests and generating events based on the responses. This Collector provides multiple Discover types and Health Check options.

Configuring a Health Check Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Health Check** from the **Manage Sources** page's tiles or left nav. Click **New Collector** to open the **Health Check** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs, some of which contain collapsible accordions. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

See Formatting Expressions below for guidance on this Collector's specific requirement to enter JavaScript expressions as template literals.

Collector Sources cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how Health Checks are made before processing.

Collector ID: Unique ID for this Collector. E.g., `healthMonitoring`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Discover

Within the **Discover** accordion, the **Discover type** drop-down provides four options, corresponding to different use cases. Each **Discover type** selection will expose a different set of fields. Below, we cover the **Discover type**s from the simplest to the most complex.

- Discover type: None matches cases where one simple API call will retrieve all the data you need. This default option suppresses the Discover stage. Use the modal's **Health check** section to specify the endpoint and other details. Cribl Stream will assign the Health Check task to a single Worker.
- Discover type: Item List matches cases where you want to enumerate a known list of **Discover items** to retrieve, like specific AWS services to identify hosts to check. Discovery will return one Health Check task per item in the list, and Cribl Stream will spread each of those tasks across all available Workers.

- **Discover type: JSON Response** provides a **Discover result** field where you can (optionally) define Discover tasks as a JSON array of objects. Each entry returned by Discover will generate a Health Check task, and Cribl Stream will spread each of those tasks across all available Workers.
- **Discover type: HTTP Request** matches cases where you need to dynamically discover what you can Health Check from a REST endpoint. This Discover type most fully exploits Cribl Stream's Discover-and-then-health-check architecture. (Example: Make a REST call to get a list of available hosts, then run Health Checks against each of those hosts.) Each item returned will generate a Health Check task, and Cribl Stream will spread each of those tasks across all available Workers.

Discover Type: Item List

Setting the **Discover type** to `Item List` exposes this additional field above the Health Check settings:

Discover items: List of items to return from the Discover task. Each returned item will generate a Health Check task, and can be referenced using `${id}` in the **Health Check URL**, the **Health Check parameters**, or the **Health Check headers**. Sample list with Cribl's health endpoint:

```
http://localhost:8080/services/collector/health,http://0.0.0.0:3000/services/collec
```

Discover Type: JSON Response

Setting the **Discover type** to `JSON Response` exposes these additional fields above the Health Check settings:

Discover result: Allows hard-coding the Discover result. Must be a JSON object. Works with the **Discover data field**. Sample JSON entry:

```
 `[ {"url": "http://localhost:8088/services/collector/health"},{ "url": "http://loc
```

Discover data field: Within the response, this is the name of the field that contains discovery results. (Leave blank if the result is an array.) Sample JSON entry:

```
items, json: { items: [{id: 'first'},{id: 'second'}] }
```

In an XML response, this is the name of the element that contains discovery results. Sample XML entry:

```
result.items
<response>
 <items>
  <element>
   <id>first</id>
  </element>
  <element>
   <id>second</id>
  </element>
 </items>
</response>
```

Discover Type: HTTP Request

Setting the **Discover type** to `HTTP Request` exposes these additional fields above the Health Check settings:

Discover URL: Enter the URL to use for the Discover operation. This can be a constant URL, or a JavaScript expression to derive the URL.

> Where a URL (path or parameters) includes variables that might contain unsafe ASCII characters, encode these variables using `C.Encode.uri(paramName)`. (Examples of unsafe characters are: space, `$`, `/`, `=`.) Example URL with encoding: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`.
>
> Request parameters that are not contained directly in the URL are automatically encoded. Cribl Stream URLs/expressions specified in the **Discover URL** field follow redirects.

Discover method: Select the HTTP verb to use for the Discover operation – `GET`, `POST`, or `POST with body`.

Discover POST body: Template for POST body to send with the Discover request. (This field is displayed only when you set the **Discover method** to `POST with body`.)

Discover parameters: Optional HTTP request parameters to append to the Discover request URL. These refine or narrow the request. Click **Add Parameter** to add parameters as key-value pairs:

- **Name**: Parameter name.

- **Value**: JavaScript expression to compute the parameter's value, normally enclosed in backticks (e.g., `` `${id}` ``). Can also be a constant, enclosed in single quotes (`'id'`). Values without delimiters (e.g., `id`) are evaluated as strings.

Discover headers: Optional Discover request headers.: Click **Add Header** to add headers as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value, normally enclosed in backticks (e.g., `` `${id}` ``). Can also be a constant, enclosed in single quotes (`'id'`). Values without delimiters (e.g., `id`) are evaluated as strings.

Discover data field: Within the response JSON, name of the field that contains Discover results. Leave blank if the result is an array.

The following sections describe the remaining tabs, whose settings and content apply equally to all **Discover type** selections.

Health Check

Each request that this Collector sends constitutes one "health check." One source of insight into the health of the endpoint is the internal fields that the Collector emits. For example, to observe how slowly or quickly an endpoint is responding, you can monitor the value of the `elapsedMS` element nested in the `__collectStats` internal field.

Health check URL: URL (constant or JavaScript expression) to use for the Health Check operation.

Where a URL (path or parameters) includes variables that might contain unsafe ASCII characters, encode these variables using `C.Encode.uri(paramName)`. (Examples of unsafe characters are: space, `$`, `/`, `=`.) Example URL with encoding: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`.

Request parameters that are not contained directly in the URL are automatically encoded. Cribl Stream URLs/expressions specified in the **Health check URL** field follow redirects.

Health check method: Select the HTTP verb to use for the Health Check operation – `GET`, `POST`, or `POST with body`.

Health check POST body: Template for POST body to send with the Health Check request. (This field is displayed only when you set the **Health check method** to `POST with body`.) You can reference parameters from the Discover response using template params of the form: `` `${variable}` ``.

Health check parameters: Optional HTTP request parameters to append to the request URL. These refine or narrow the request. Click **Add parameter** to add parameters as key-value pairs:

- **Name**: Field name.
- **Value**: JavaScript expression to compute the field's value, normally enclosed in backticks (e.g., `` `${id}` ``). Can also be a constant, enclosed in single quotes (`'id'`). Values without delimiters (e.g., `id`) are evaluated as strings.

Health check headers: Click **Add Header** to (optionally) add request headers as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value, normally enclosed in backticks (e.g., `` `${id}` ``). Can also be a constant, enclosed in single quotes (`'id'`). Values without delimiters (e.g., `id`) are evaluated as strings.

By adding the appropriate **Health check headers**, you can specify authentication based on API keys, as an alternative to the **Authentication**: `Basic` or `Login` options below.

Authenticate health check: Toggle to `Yes` if you want to authenticate Health Checks. When set to `No`, the default, **Authentication** is applied only to Discover.

Authentication

In the **Authentication** accordion, use the **Authentication** drop-down to select one of these options:

- **None**: Don't use authentication. Compatible with REST servers like AWS, where you embed a secret directly in the request URL.
- **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.
- **Login**: Enables you to specify several credentials, then perform a POST to an endpoint during the Discover operation. The POST response returns a token, which Cribl Stream uses for later Health Check operations. Exposes multiple extra fields – see Login Authentication below.
- **Login (credentials secret)**: Like **Login**, except that you specify a secret referencing the credentials, rather than the credentials themselves. Exposes multiple extra fields – see Login Secret Authentication below.
- **OAuth**: Enables you to directly enter credentials for authentication via the OAuth protocol. Exposes multiple extra fields – see OAuth Authentication below.

- **OAuth (text secret)**: Like the **OAuth** option, except that you specify a stored secret referencing the credentials. Exposes multiple extra fields – see OAuth Secret Authentication below.

Login Authentication

Selecting `Login` exposes the following additional fields:

- **Login URL**: URL for the login API call, which is expected to be a POST call.
- **Username**: Login username.
- **Password**: Login password.
- **POST body**: Template for POST body to send with the login request. The `${username}` and `${password}` variables specify the corresponding credentials' locations in the message.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in Discover and Health Check calls. Uses `${token}` to reference the token obtained from the login POST request.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – e.g., `id` instead of `` `${id}` `` – will be evaluated as strings.

Login Secret Authentication

Selecting **Login (credentials secret)** exposes the following additional fields:

- **Login URL**: URL for the login API call, which is expected to be a POST call.
- **Credentials secret**: Select a stored text secret in this drop-down, or click **Create** to configure a new secret.
- **POST body**: Template for POST body to send with the login request. The `${username}` and `${password}` variables specify the corresponding credentials' locations in the message.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in Discover and Health Check calls. Uses `${token}` to reference the token obtained from the login POST request.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row'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.

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.
- **Client secret parameter**: The name of the parameter to send with the **Client secret value**.
- **Client secret value**: The OAuth access token to authorize requests.
- **Extra authentication parameters**: Optionally, click **Add parameter** for each additional OAuth request parameter you want to send in the body of POST requests. Automatically sets the `Content-Type` header to `application/x-www-form-urlencoded`.
  In the resulting table, enter each parameter row'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 `` `${id}` `` – will be evaluated as strings.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in Discover and Health Check calls. Uses `${token}` to reference the token obtained from the login POST request.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row'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.

OAuth Secret Authentication

Selecting **OAuth (text secret)** exposes the following additional fields:

- **Login URL**: Endpoint for the OAuth API call, which is expected to be a POST call.
- **Client secret parameter**: The name of the parameter to send with the **Client secret value**.
- **Client secret value (text secret)**: Select a stored text secret in this drop-down, or click **Create** to configure a new secret.
- **Extra authentication parameters**: Optionally, click **Add parameter** for each additional OAuth request parameter you want to send in the body of POST requests. Automatically sets the `Content-Type` header to `application/x-www-form-urlencoded`.

In the resulting table, enter each parameter row'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.

- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in Discover and Health Check calls. Uses `${token}` to reference the token obtained from the login POST request.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – e.g., `id` instead of `` `${id}` `` – will be evaluated as strings.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Optional Settings

Request Timeout (secs): Here, you can set a maximum time period (in seconds) for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `0`, which disables timeout metering.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (e.g., self-signed certificates). Defaults to `No`.

Safe headers: Optionally, list headers that you consider safe to log in plain text. Separate the header names with tabs or hard returns.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the Health Check data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. Defaults to `Cribl`.

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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination.

The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. One use case might be a Health Check Collector that gathers a known, simple type of data from a single endpoint. This approach keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields: List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc Health Check jobs if Cribl Stream restarts during the jobs' execution.

Formatting Expressions

JavaScript expression fields in this Health Check Collector behave differently from those elsewhere in Cribl Stream. Here, you must enter expressions as template literals, with placeholders referencing variables and JS functions. Here are a few examples:

- To reference the `__collectible.id` internal field in a URL, header, or parameter (etc.), you would write the variable as ``` `${id}` ```.
- To reference a function call: ``` `${Date.now()}` ```.
- To call a function that references a variable: ``` `${Math.floor(earliest)}` ```.

Responses

The Health Check Collector generates error or success events based on the received response from the Health Check request.

Error Events

The Health Check Collector generates error events when no response is received, a request times out, or receives an error (non 200) response. The following is an example error event:

```
{
  "host": "Cribl.local",
  "_raw": "{\"url\":\"http://0.0.0.0:3000/services/collector/health\",\"method\":\'
  "method": "get",
  "url": "http://0.0.0.0:3000/services/collector/health",
  "statusCode": 404,
  "error": "Http Error, statusCode: 404, details: {\"host\":\"0.0.0.0:3000\",\"port
  "_time": 1660152577.93,
  "cribl_breaker": "Cribl:ndjson"
}
```

Success Events

The Health Check Collector generates success events when a 200 success status code is received. The following is an example success event:

```
{
  "host": "Cribl.local",
  "_raw": "{\"url\":\"http://0.0.0.0:3000/api/v1/health\",\"method\":\"get\",\"stat
  "method": "get",
  "url": "http://0.0.0.0:3000/api/v1/health",
  "statusCode": 200,
  "body": "{\"status\":\"healthy\",\"startTime\":1660152114781}",
  "_time": 1660152114.781,
  "cribl_breaker": "Cribl:ndjson"
}
```

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These “meta” fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object's nested fields contain metadata about each collection job.
- `__collectStats` – This object's nested fields are:
    - `method`: The value you set for **Health Check** > **Health check method**.
    - `url`: The value you set for **Health Check** > **Health check URL**.
    - `elapsedMS`: The time elapsed from the moment the Health Check request was sent until the response was received.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a "stuck", "orphaned," or "zombie" job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream's logs.

To keep stuck jobs from running excessively long:

- First, try setting **Optional Settings** > **Request Timeout (secs)** – whose default of `0` means "wait forever" – to a desired maximum duration.
- If adjusting **Timeout (secs)** does not fix the problem, try the global setting **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration. You'll find **Job Timeout** among the task manifest and buffering limits.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

15.1.7. REST / API Endpoint

Cribl Stream supports collecting data from REST endpoints. This Collector provides multiple Discover types and Collect options.

For usage examples, see Using REST / API Collectors and our adjacent guides. To configure a Collector for Netskope Events and Alerts, see this Netskope Community topic.

Configuring a REST Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **REST** > **New Collector** modal, which provides the following options and fields.

Alternatively, you can create a REST Collector by importing its configuration and entering desired values for any placeholders.

Whichever method you use, click **Save** once you have configured your Collector.

The sections described below are spread across several tabs, some of which contain collapsible accordions. Click the tab links at left to navigate among tabs.

See Formatting Expressions, below, for guidance on this Collector's specific requirement to enter JavaScript expressions as template literals.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. For example, `rest42json`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Discover Settings

Within the **Discover** accordion, the **Discover type** drop-down provides four options, corresponding to different use cases. Each **Discover type** selection will expose a different set of **Collector Settings** fields.

Below, we cover the **Discover type**s from simplest to most-complex.

- Discover type: None matches cases where one simple API call will retrieve all the data you need. This default option suppresses the Discover stage. Use the modal's **Collect** section to specify the endpoint and other details. Cribl Stream will assign the Collect task to a single Worker. (Example: Collect a list of configured Cribl Stream Pipelines.)
- Discover type: Item List matches cases where you want to enumerate a known list of **Discover items** to retrieve. (Examples: Collect network traffic data that's tagged with specific subnets; or collect weather data for a known list of ZIP codes.) Discovery will return one Collect task per item in the list, and Cribl Stream will spread each of those Collect tasks across all available Workers.
- Discover type: JSON Response provides a **Discover result** field where you can (optionally) define Discover tasks as a JSON array of objects. Each entry returned by Discover will generate a Collect task, and Cribl Stream will spread each of those Collect tasks across all available Workers. (Example: Collect data for specific geo locations in the National Weather Service API's stream of worldwide weather data. This particular API requires multiple parameters in the request URL – latitude, longitude, and so forth – so **Item List** discovery would not work.)
- Discover type: HTTP Request matches cases where you need to dynamically discover what you can collect from a REST endpoint. This Discover type most fully exploits Cribl Stream's Discover-and-then-Collect architecture. (Example: Make a REST call to get a list of available log files, then run Collect against each of those files.) Each item returned will generate a Collect task, and Cribl Stream will spread each of those Collect tasks across all available Workers. As of Cribl Stream 3.0.2, this Discover type supports XML responses.

Collect Settings (Common)

One source of insight about Collect requests and the behavior of the endpoints where they are sent is the internal fields that the Collector emits. For example, to observe how slowly or quickly an endpoint is responding, you can monitor the value of the `elapsedMS` element nested in the `__collectStats` internal field.

Within the **Collect** accordion, the following options appear for **Discover type**: `None`, as well as for all other **Discover type** selections:

Collect URL: URL (constant or JavaScript expression) to use for the Collect operation. This URL can use an `authToken` variable to access an auth token. For example, if `1234567890` is defined as an auth token and you use `http://myurl.com?authToken=${authToken}` as the Collect URL, the Collect operation will use `http://myurl.com?authToken=1234567890`.

Where a URL (path or parameters) includes variables that might contain unsafe ASCII characters, encode these variables using `C.Encode.uri(paramName)`. (Examples of unsafe characters are:

space, `$`, `/`, `=`.) Example URL with encoding: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`.

Request parameters are not contained directly in the URL are automatically encoded. Cribl Stream URLs/expressions specified in the **Collect URL** field follow redirects.

Collect method: Select the HTTP verb to use for the Collect operation – `GET`, `POST`, `POST with body`, or `Other`.

Collect POST body: Template for POST body to send with the Collect request. (This field is displayed only when you set the **Collect method** to `POST with body`.) You can reference parameters from the Discover response using template params of the form: `` `${variable}` ``. This field can reference the auth token variable obtained during authentication, using the variable `authToken`. For example: `` `{ "token": ${authToken}, "param1": "value1" }` ``.

Collect verb: Custom HTTP method to use for the Collect operation. (This field is displayed only when you set the **Collect method** to `Other`.)

Collect parameters: Optional HTTP request parameters to append to the request URL. These refine or narrow the request. Click **Add Parameter** to add parameters as key-value pairs:

- **Name**: Field name.
- **Value**: JavaScript expression to compute the field's value, normally enclosed in backticks (for example, `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (for example, `earliest`) are evaluated as strings.

Collect headers: Click **Add Header** to (optionally) add collection request headers as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value, normally enclosed in single quotes. For example: `'someConstantValue'`. Values without delimiters (for example, `someConstantValue`) are evaluated as strings.

By adding the appropriate **Collect headers**, you can specify authentication based on API keys, as an alternative to the **Authentication**: `Basic` or `Login` options below.

Pagination: For the options exposed by this drop-down list, see the Pagination section below.

Authentication: For the options within this accordion, see the Authentication Settings section below.

Time Range Variables

> The following fields accept `` `${earliest}` `` and `` `${latest}` `` variables, which reference any **Time Range** values that have been set in manual or scheduled collection jobs:
>
> - **Collect URL**, **Collect parameters**, **Collect headers**
> - **Discover URL**, **Discover parameters**, **Discover headers**.
>
> As an example, here is a **Collect URL** entry using these variables:
> `http://localhost/path?from=${earliest}&to=${latest}`
>
> Both variables are formatted as UNIX epoch time, in seconds units. When using them in contexts that require milliseconds resolution, multiply them by 1,000 to convert to ms.

Pagination

Use this drop-down list to select the pagination scheme for collection results. Defaults to `None`. The other options, expanded below, are:

- Response Body Attribute / Response Header Attribute
- RFC 5988 – Web Linking
- Offset/Limit
- Page/Size

Response Body Attribute / Response Header Attribute

Select `Response Body Attribute` to extract a value from the response body that identifies the next page of data to retrieve.

Select `Response Header Attribute` to extract this next-page value from the response header. Either of these selections exposes these additional fields:

- **Response attribute:** Name(s) of attribute(s) in the response payload or header that contain next-page information. If you have multiple attributes with the same name within the response body, you can provide the path to the one you need, for example: `foo.0.bar`. If Cribl Stream cannot find an exact match for a specified attribute name, it will fallback to the last-found partial match.
- **Max pages**: The maximum number of pages to retrieve per Collect task. Defaults to `50` pages. Set to `0` to retrieve all pages.

Selecting `Response Body Attribute` exposes one more field:

- **Last-page expression**: Optionally, enter a JavaScript expression that Cribl Stream will use to determine when the last page has been reached. The values tested by this expression must be present in the

Response attributes section above. Example expression: `morePages === false`.

Use this expression with APIs like Smartsheet Event Reporting, which return an attribute that signals whether more pages are present, rather than simply returning an empty result set beyond the last page.

RFC 5988 – Web Linking

Select this option with APIs that follow RFC 5988 conventions to provide the next-page link in a header. This selection exposes three additional fields:

Next page relation name: Header substring that refers to the next page in the result set. Defaults to `next`, corresponding to the following example link header:
`<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"`

Current page relation name: Optionally, specify the relation name within the link header that refers to the current result set. In this same example, `rel="self"` refers to the current page of results:
`<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"`

Max pages: The maximum number of pages to retrieve. Defaults to `50` pages. Set to `0` to retrieve all pages.

Offset/Limit

Select this option to receive data from APIs that use offset/limit pagination. This selection exposes several additional fields:

Offset field name: Query string parameter that sets the index from which to begin returning records. For example: `/api/v1/query?term=cribl&limit=100&offset=0`. Defaults to `offset`.

Starting offset: (Optional) offset index from which to start request. Defaults to undefined, which will start collection from the first record.

Limit field name: Query string parameter to set the number of records retrieved per request. For example: `/api/v1/query?term=cribl&limit=100&offset=0`. Defaults to `limit`.

Limit: Maximum number of records to collect per request. Defaults to `50`.

Total record count field name: Identifies the attribute, within the response, that contains the total number of records for the query.

Max pages: The maximum number of pages to retrieve. Defaults to `50`. Set to `0` to retrieve all pages.

Zero-based index: Toggle to `Yes` to indicate that the requested data's first record is at index `0`. The default (No) indicates that the first record is at index `1`.

Page/Size

Select this option to receive data from APIs that use page/size pagination. This selection exposes several additional fields:

Page number field name: Query string parameter that sets the page index to be returned. For example: `/api/v1/query?term=cribl&page_size=100&page_number=0`. Defaults to `page`.

Starting page number: (Optional) page number from which to start request. Defaults to undefined, which will start collection from the first page.

Page size field name: Query string parameter to set the number of records retrieved per request. For example:
`/api/v1/query?term=cribl&page_size=100&page_number=0`. Defaults to `size`.

Page size: Maximum number of records to collect per page. Defaults to `50`.

Total page count field name: Identifies the attribute, within the response, that contains the total number of pages for the query.

Total record count field name: Identifies the attribute, within the response, that contains the total number of records for the query.

Max pages: The maximum number of pages to retrieve. Defaults to `50`. Set to `0` to retrieve all pages.

Zero-based index: Toggle to `Yes` to indicate that the requested data's first record is at index `0`. The default (No) indicates that the first record is at index `1`.

Authentication Settings

In the **Authentication** accordion, use the **Authentication** drop-down to select one of these options:

- **None**: Don't use authentication. Compatible with REST servers like AWS, where you embed a secret directly in the request URL.
- **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.

- **Login**: Enables you to specify several credentials, then perform a POST to an endpoint during the Discover operation. The POST response returns a token, which Cribl Stream uses for later Collect operations. Exposes multiple extra fields – see Login Authentication below.
- **Login (credentials secret)**: Like **Login**, except that you specify a secret referencing the credentials, rather than the credentials themselves. Exposes multiple extra fields – see Login Secret Authentication below.
- **OAuth**: Enables you to directly enter credentials for authentication via the OAuth protocol. Exposes multiple extra fields – see OAuth Authentication below.
- **OAuth (text secret)**: Like the **OAuth** option, except that you specify a stored secret referencing the credentials. Exposes multiple extra fields – see OAuth Secret Authentication below.

Login Authentication

Login is the best authentication type to use for all services that return a JSON web token (JWT) or other authentication token whose content type is `application/json` or `content/plain`.

Selecting **Login** exposes the following additional fields:

- **Login URL**: URL for the login API call, which is expected to be a POST call.
- **Username**: Login username.
- **Password**: Login password.
- **POST body**: Template for POST body to send with the login request. The `${username}` and `${password}` variables specify the corresponding credentials' locations in the message.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes. If the response from the auth request provides the JWT as `plain/text`, leave this field blank.
- **Authorization header**: Authorization header key to pass in Discover and Collect calls. Defaults to the literal name `Authorization`. (Cribl Stream 4.0 and later.)
- **Authorize expression**: JavaScript expression used to compute an Authorization header to pass in Discover and Collect calls. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request. If you need to pass the token in a Collector or Discover request parameter, URL, or request body, you can access it with `${authToken}`.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

Login Secret Authentication

Login is the best authentication type to use for all services that return a JSON web token (JWT) or other authentication token whose content type is `application/json` or `content/plain`.

Selecting **Login (credentials secret)** exposes the following additional fields:

- **Login URL**: URL for the login API call, which is expected to be a POST call.
- **Credentials secret**: Select a stored text secret in this drop-down, or click **Create** to configure a new secret.
- **POST body**: Template for POST body to send with the login request. The `${username}` and `${password}` variables specify the corresponding credentials' locations in the message.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes. If the response from the auth request provides the JWT as `plain/text`, leave this field blank.
- **Authorization header**: Authorization header key to pass in Discover and Collect calls. Defaults to the literal name `Authorization`. (Cribl Stream 4.0 and later.)
- **Authorize expression**: JavaScript expression used to compute an Authorization header to pass in Discover and Collect calls. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request. If you need to pass the token in a Collector or Discover request parameter, URL, or request body, you can access it with `${authToken}`.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

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.
- **Client secret parameter**: The name of the parameter to send with the **Client secret value**.
- **Client secret value**: The OAuth access token to authorize requests.
- **Extra authentication parameters**: Optionally, click **Add parameter** for each additional OAuth request parameter you want to send in the body of POST requests. Automatically sets the `Content-Type` header to `application/x-www-form-urlencoded`.
  In the resulting table, enter each parameter row's **Name**. The corresponding **Value** is a JavaScript expression to compute the parameter's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.

- **Authorization header**: Authorization header key to pass in Discover and Collect calls. Defaults to the literal name `Authorization`. (Cribl Stream 4.0 and later.)
- **Authorize expression**: JavaScript expression used to compute an Authorization header to pass in Discover and Collect calls. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request. If you need to pass the token in a Collector or Discover request parameter, URL, or request body, you can access it with `${authToken}`.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

OAuth Secret Authentication

Selecting **OAuth (text secret)** exposes the following additional fields:

- **Login URL**: Endpoint for the OAuth API call, which is expected to be a POST call.
- **Client secret parameter**: The name of the parameter to send with the **Client secret value**.
- **Client secret value (text secret):** Select a stored text secret in this drop-down, or click **Create** to configure a new secret.
- **Extra authentication parameters**: Optionally, click **Add parameter** for each additional OAuth request parameter you want to send in the body of POST requests. Automatically sets the `Content-Type` header to `application/x-www-form-urlencoded`.
  In the resulting table, enter each parameter row's **Name**. The corresponding **Value** is a JavaScript expression to compute the parameter's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorization header**: Authorization header key to pass in Discover and Collect calls. Defaults to the literal name `Authorization`. (Cribl Stream 4.0 and later.)
- **Authorize expression**: JavaScript expression used to compute an Authorization header to pass in Discover and Collect calls. Uses `${token}` to reference the token obtained from a Login or OAuth authorization request. If you need to pass the token in a Collector or Discover request parameter, URL, or request body, you can access it with `${authToken}`.
- **Authentication headers**: Optionally, click **Add Header** for each custom auth header you want to define. In the resulting table, enter each header row's **Name**.
  The corresponding **Value** is a JavaScript expression to compute the header's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

Google Service Account OAuth Authentication

- **Scopes**: OAuth 2.0 scopes used to request access to Google APIs during authentication. See the Google page OAuth 2.0 Scopes for more information. Click **Add Scope** to add a scope.
- **Service account credentials**: Contents of the Google Cloud service account credentials (JSON keys) file. To upload a file, click the upload icon in this field's upper right corner.
- **Impersonated account's email address**: Email address of a user account with super admin permissions to the resources the Collector will retrieve.

Google Service Account OAuth Secret Authentication

- **Scopes**: OAuth 2.0 scopes used to request access to Google APIs during authentication. See the Google page OAuth 2.0 Scopes for more information. Click **Add Scope** to add a scope.
- **Service account credentials (text secret)**: Text secret containing the value of the Google service account credentials. Select an existing text secret or click **Create** to create a new secret.
- **Impersonated account's email address**: Email address of a user account with super admin permissions to the resources the Collector will retrieve.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so forth.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Optional Settings

Request Timeout (secs): Here, you can set a maximum time period (in seconds) for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `0`, which disables timeout metering.

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.

Disable time filter: Toggle to `Yes` if your Run or Schedule configuration specifies a date range and no events are being collected. This will disable the Collector's event time filtering to prevent timestamp conflicts.

Decode URL: When toggled to `No`, Cribl Stream will not decode URLs before sending the HTTP request. This setting applies to collect, authentication, and pagination requests. Defaults to `Yes`.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to `No`.

Safe headers: Optionally, list headers that you consider safe to log in plain text. Separate the header names with tabs or hard returns. These headers will not be redacted when exporting the Collector configuration as JSON.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Settings By Discover Type

For Discover types other than `None`, Cribl Stream exposes additional fields specific to the selected Discover type.

Discover Type: Item List

Setting the **Discover type** to `Item List` exposes this additional field above the Common Collector Settings:

Discover items: List of items to return from the Discover task. Each returned item will generate a Collect task, and can be referenced using `${id}` in the **Collect URL**, the **Collect parameters**, or the **Collect headers**.

Discover Type: JSON Response

Setting the **Discover type** to `JSON Response` exposes these additional fields above the Common Collector Settings:

Discover result: Allows hard-coding the Discover result. Must be a JSON object. Works with the Discover data field.

Discover data field: Within the response, this is the name of the field that contains discovery results. (Leave blank if the result is an array.) Sample JSON entry:

```
items, json: { items: [{id: 'first'},{id: 'second'}] }
```

In an XML response, this is the name of the element that contains discovery results. Sample XML entry:

```
result.items
<response>
 <items>
  <element>
   <id>first</id>
  </element>
  <element>
   <id>second</id>
  </element>
 </items>
</response>
```

Discover Type: HTTP Request

Setting the **Discover type** to `HTTP Request` exposes these additional fields above the Common Collector Settings:

Discover URL: Enter the URL to use for the Discover operation. This can be a constant URL, or a JavaScript expression to derive the URL.

Where a URL (path or parameters) includes variables that might contain unsafe ASCII characters, encode these variables using `C.Encode.uri(paramName)`. (Examples of unsafe characters are: space, `$`, `/`, `=`.) Example URL with encoding: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`.

Request parameters are not contained directly in the URL are automatically encoded. Cribl Stream URLs/expressions specified in the **Discover URL** field follow redirects.

Discover method: Select the HTTP verb to use for the Discover operation – `GET`, `POST`, `POST with body`, or `Other`.

Discover POST body: Template for POST body to send with the Discover request. (This field is displayed only when you set the **Discover method** to `POST with body`.)

Discover verb: Custom HTTP method to use for the Discover operation. (This field is displayed only when you set the **Discover method** to `Other`.)

Discover parameters: Optional HTTP request parameters to append to the Discover request URL. These refine or narrow the request. Click **Add Parameter** to add parameters as key-value pairs:

- **Name**: Parameter name.
- **Value**: JavaScript expression to compute the parameter's value, normally enclosed in backticks (for example, `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (for example, `earliest`) are evaluated as strings.

Discover headers: Optional Discover request headers.: Click **Add Header** to add headers as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value, normally enclosed in backticks (for example, `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (for example, `earliest`) are evaluated as strings.

Discover data field: Within the response JSON, name of the field that contains Discover results. Leave blank if the result is an array.

The following sections describe the Collector Settings' remaining tabs, whose settings and content apply equally to all **Discover type** selections.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field’s value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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`, and so forth. (Example: `42 MB`.) Default value of `0` indicates no throttling.

> You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. One use case might be a REST Collector that gathers a known, simple type of data from a single endpoint. This approach keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Pre-processing Pipeline: Pipeline to process results before sending to Routes. Optional, and available only when **Send to Routes** is toggled to `Yes`.

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`, and so forth. (Example: `42 MB`.) Default value of `0` indicates no throttling.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Working with State Tracking

You can configure the Collector to track state, either by time or another arbitrary value. This can help prevent overlaps between jobs, where subsequent Collector runs may return some of the same results as previous runs. Similarly, it can help prevent gaps in data by allowing a Collector run to pick up from where the last run ended.

State tracking can be particularly useful for tracking the greatest `_time` value for events returned from a collection run, but you can also use it to derive arbitrary state for custom use cases. You can access saved state values in any Collector field that supports JavaScript expressions.

For a hands-on example of setting up a Collector with state tracking enabled, see Using REST/API Collectors.

Tracking State for Collector Runs

You can enable state tracking under the `State Tracking` section when scheduling or performing a full run of the Collector (Preview and Discover runs do not support state tracking). Once enabled, two more fields will appear for state tracking configuration.

State update expression: JavaScript expression that defines how to update the state from an event. Use the event's data and the current state to compute the new state.

State merge expression: JavaScript expression that defines which state to keep when merging a task's newly reported state with the previously saved state. Evaluates `prevState` and `newState` variables, resolving to the state to keep.

The default values for these fields are configured to track state by the latest `_time` field found in events gathered in a collection run.

Understanding State Expression Fields

The **State update** and **State merge** expressions control how state is derived from a collection run and how it is merged with existing state, respectively. They're preconfigured to work with the common use case of tracking state by latest `_time`, but you may need to update them for other use cases. To understand what these fields do, let's break down the default values.

State update expression

This expression has a default value of:

```
__timestampExtracted !== false && {latestTime: (state.latestTime || 0) > _time ? st
```

__timestampExtracted !== false - the `__timestampExtracted` field is set to false if the Event Breaker was unable to parse time for the event. If this is the case, we don't want to update state (the event's `_time` value defaults to `Date.now()` if the Event Breaker was unable to parse out the correct time). If `_timestampExtracted` is `false`, we take advantage of short-circuit evaluation to not update state.

State values must resolve to an object, such as:

```
{ "latestTime": 17122806161 }
```

If the expression does not resolve to an object, Cribl Stream will ignore the result.

{latestTime: (state.latestTime || 0) > _time ? state.latestTime : _time} - compare `state.latestTime` to the event's `_time` value, keeping whichever value is greater.

State merge expression

This expression has a default value of:

```
prevState.latestTime > newState.latestTime ? prevState : newState
```

It compares `prevState` (the state that was previously saved) to `newState` (the state reported from the most recent collection task), keeping the state with the greatest `latestTime` value.

Managing Collector State

Click **Manage State** to view, modify, or delete Collector state. For more information, see Manage Collector State.

Importing a Collector Configuration

You can create a REST Collector (or any other Collector) by importing a JSON file that defines a Collector configuration, entering desired values for any placeholders, and saving the new Collector.

Creating a New Collector from a Template explains how to do this, and Cribl's Collector Templates repository provides configurations for many popular Collectors (with companion Event Breakers, and event samples in some cases). You can import these REST Collectors and save the time you'd have spent building them yourself.

Formatting Expressions

JavaScript expression fields in this REST/API Collector behave differently from those elsewhere in Cribl Stream. Here, you must enter expressions as template literals, with placeholders referencing variables and JS functions. Here are a few examples:

- To reference the earliest time variable in a URL, header, or parameter (and so forth), you would write the variable as `` `${earliest}` ``.
- To reference a function call: `` `${Date.now()` ``.
- To call a function that references a variable: `` `${Math.floor(earliest)}` ``.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These “meta” fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object’s nested fields contain metadata about each collection job.
- `__collectStats` – This object’s nested fields are:
  - `method`: The value you set for **Collect** > **Collect method**.
  - `url`: The value you set for **Collect Settings (Common)** > **Collect URL**.
  - `elapsedMS`: The time elapsed from the moment the Collector’s request was sent until the response was received.

Response Errors

The Collector treats all non-`200` responses from configured URL endpoints as errors. This includes `1xx`, `3xx`, `4xx`, and `5xx` responses.

On Discovery, Preview, and most Collect jobs, it interprets these as fatal errors. On Collect jobs, a few exceptions are treated as non-fatal:

- Where a Collect job launches multiple tasks, and only a subset of those tasks fail, Cribl Stream places the job in failed status, but treats the error as non-fatal. (Note that Cribl Stream does not retry the failed tasks.)

- Where a Collect job receives a `3xx` redirection error code, it follows the error's treatment by the underlying library, and does not necessarily treat the error as fatal.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a "stuck", "orphaned," or "zombie" job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream's logs.

To keep stuck jobs from running excessively long:

- First, try setting **Optional Settings** > **Request Timeout (secs)** – whose default of `0` means "wait forever" – to a desired maximum duration.
- If adjusting **Timeout (secs)** does not fix the problem, try the global setting **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration. You'll find **Job Timeout** among the task manifest and buffering limits.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

15.1.8. S3

Cribl Stream supports collecting data from Amazon S3 stores. This page covers how to configure the Collector.

For a step-by-step tutorial on using Cribl Stream to replay data from an S3-compatible store, see our Data Collection & Replay sandbox. The sandbox takes about 30 minutes. It provides a hosted environment, with all inputs and outputs preconfigured for you.

Also see our Amazon S3 Better Practices and Using S3 Storage and Replay guides.

How the Collector Pulls Data

When you run an S3 Collector in Discovery mode, the first available Worker returns the list of available files to the Leader Node.

In Full Run mode, the Leader distributes the list of files to process across 1..*N* Workers, as evenly as possible, based on file size. Each Worker then streams the files from the S3 bucket/path to itself.

Compression

Cribl Stream can ingest compressed S3 files if they meet all the following conditions:

- Compressed with the `x-gzip` MIME type.
- End with the `.gz` extension.
- Can be uncompressed using the `zlib.gunzip` algorithm.

Storage Class Compatibility

Cribl Stream does **not** support data preview, collection, or replay from S3 Glacier or S3 Deep Glacier storage classes, whose stated retrieval lags (variously minutes to 48 hours) cannot guarantee data availability when the Collector needs it.

Cribl Stream **does** support data preview, collection, and replay from S3 Glacier Instant Retrieval when you're using the S3 Intelligent-Tiering storage class.

Configuring an S3 Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **S3** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **S3** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. For example, `Attic42TreasureChest`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Auto-populate from: Select a Destination with which to auto-populate Collector settings. Useful when replaying data.

S3 bucket: Simple Storage Service bucket from which to collect data.

Authentication

Use the **Authentication Method** drop-down to select an AWS authentication method.

The **Auto** option (default) will use environment variables `AWS_ACCESS_KEY_ID` and `AWS_SECRET_ACCESS_KEY`, or the attached IAM role. Will work only when running on AWS.

The **Manual** option exposes these 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. This value can be a constant or a JavaScript expression.
- **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. Optional when running on AWS. This value can be a constant or a JavaScript expression.

The **Secret** option swaps in this drop-down:

- **Secret key pair**: Select a secret key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS. Follow the **Create** link if you need to configure a key pair.

The **Manual** option's **Access key** and **Secret key** values override the **Auto** option. For any Workers that you want to associate with an AWS IAM role, using the **Assume Role** controls below – including Cribl-managed or hybrid Cribl.Cloud Workers – ensure that the **Manual** option's key fields are empty.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable Assume Role: Slide to `Yes` to enable Assume Role behavior.

AssumeRole ARN: Amazon Resource Name (ARN) of the role to assume.

External ID: External ID to use when assuming role. Enter this if defined in your IAM policy's trust relationship; otherwise, leave blank. (Usage example: AWS Cross-Account Data Collection.)

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).

Optional Settings

Region: S3 Region from which to retrieve data.

Path: Path, within the bucket, from which to collect data. Templates and tokens are supported to provide flexibility and automation in defining the directory structure, allowing you to organize data efficiently based on various parameters. This value can be a constant or a JavaScript expression. More on templates and tokens here.

- Templating allows for dynamic insertion of variables like host, year, and month. For example, `/myDir/${host}/${year}/${month}/`).
- Time-based tokens enable time-specific data organization. For example, `myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/`.

If this S3 Collector will be used to round-trip data through our Destination and replay it in Cribl Stream, we have a recommended **Path**. See the replay Collector for details.

Path extractors: Extractors enable you to use template tokens as context for expressions that enrich discovery results. For usage details and examples, see Using Path Extractors.

Endpoint: S3 service endpoint. If empty, Cribl Stream 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 Stream will automatically add it to the URL path. For details, see AWS' Path-Style Requests topic.

Signature version: Signature version to use for signing S3 requests. Defaults to `v4`.

Max batch size (objects): Maximum number of metadata objects to batch before recording as results.

Recursive: If set to `Yes` (the default), data collection will recurse through subdirectories.

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, such as self-signed certificates. Defaults to `Yes`.

Verify bucket permissions: Toggle this to `No` if you can access files within the bucket, but not the bucket itself. This resolves errors of the form: `discover task initialization failed...error: Forbidden`.

Disable time filter: Toggle to `Yes` if your Run or Schedule configuration specifies a date range and no events are being collected. This will disable the Collector's event time filtering to prevent timestamp conflicts.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained

and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Using Path Extractors

Click **Add Extractor** to add each extractor as a key-value pair, mapping a **Token** name on the left (a substring of the **Path**) to a custom JavaScript **Extractor expression** on the right. Example `{host: value.toLowerCase()}`

Each expression accesses its corresponding token through its `value` variable, and evaluates the token to populate event fields.

You can define each token in only one **Extractor expression** per S3 Collector config.

Example #1

Here is a simple, complete example, using a single expression:

Token	Extractor Expression	Matched Value	Extracted Result
`/var/log/${foobar}`	`foobar: {program: value.split('.')[0]}`	`/var/log/syslog.1`	`{program: syslog, foobar: syslog.1}`

Example #2

Here is a multi-expression example. Assuming that your S3 bucket path supports the Collector’s configured **Path**, this shows how to extract each segment of this example **Path**, and transform it into a reformatted field:

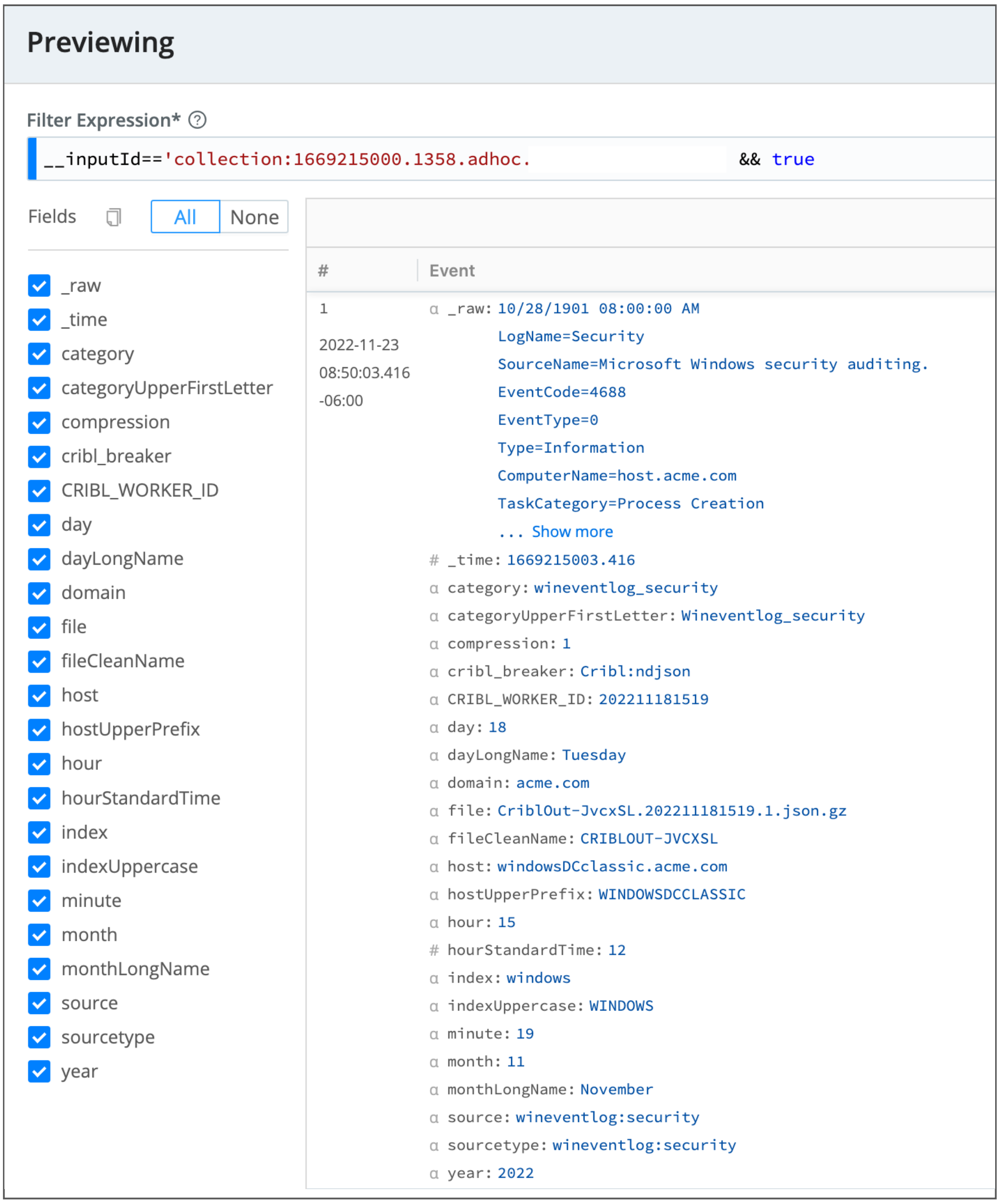
First, the path:

```
/${index}/${_time:%Y}/${_time:%m}/${_time:%d}/${host}/${category}/${_time:%H}/${file}
```

Next, the tokens and extractor expressions:

Token	Extractor Expression	Purpose
`index`	`{"indexUppercase" : value.toUpperCase()}`	Uppercase `index`
`host`	`{"hostUpperPrefix" : value.toUpperCase().replace(/(.+?)\..+/,'$1'), "domain" : value.match(/.+?\.(.+)/)[1]}`	Use one to two fields: `hostUppe` and `domai`
`category`	``{"categoryUpperFirstLetter" : `${value.substring(0,1).toUpperCase()}${value.substring(1)}`}``	Uppercase `category` first letter
`file`	`{"fileCleanName" : value.replace(/ARCHIVE-/i,'').toUpperCase().split('.')[0], "CRIBL_WORKER_ID" : value.split('.')[1], "compression" : value.split('.')[2]}`	Split the ob name) into fields

Here is a corresponding example event, as collected/replayed from S3. The metadata fields at the bottom show the results of the path extractor expressions:


Previewing

Filter Expression*

```
__inputId=='collection:1669215000.1358.adhoc.                  && true
```

Fields All None

- ✓ _raw
- ✓ _time
- ✓ category
- ✓ categoryUpperFirstLetter
- ✓ compression
- ✓ cribl_breaker
- ✓ CRIBL_WORKER_ID
- ✓ day
- ✓ dayLongName
- ✓ domain
- ✓ file
- ✓ fileCleanName
- ✓ host
- ✓ hostUpperPrefix
- ✓ hour
- ✓ hourStandardTime
- ✓ index
- ✓ indexUppercase
- ✓ minute
- ✓ month
- ✓ monthLongName
- ✓ source
- ✓ sourcetype
- ✓ year

```
#              Event
1              α _raw: 10/28/1901 08:00:00 AM
2022-11-23           LogName=Security
08:50:03.416         SourceName=Microsoft Windows security auditing.
-06:00               EventCode=4688
                     EventType=0
                     Type=Information
                     ComputerName=host.acme.com
                     TaskCategory=Process Creation
                     ... Show more
               # _time: 1669215003.416
               α category: wineventlog_security
               α categoryUpperFirstLetter: Wineventlog_security
               α compression: 1
               α cribl_breaker: Cribl:ndjson
               α CRIBL_WORKER_ID: 202211181519
               α day: 18
               α dayLongName: Tuesday
               α domain: acme.com
               α file: CriblOut-JvcxSL.202211181519.1.json.gz
               α fileCleanName: CRIBLOUT-JVCXSL
               α host: windowsDCclassic.acme.com
               α hostUpperPrefix: WINDOWSDCCLASSIC
               α hour: 15
               # hourStandardTime: 12
               α index: windows
               α indexUppercase: WINDOWS
               α minute: 19
               α month: 11
               α monthLongName: November
               α source: wineventlog:security
               α sourcetype: wineventlog:security
               α year: 2022
```

Event with path extractors' transformations

Temporary Access via SSO Provider

You can use Okta or SAML to provide access to S3 buckets using temporary security credentials.

Troubleshooting

When permissions are correct on the object store, and events are reaching the Collector, the Preview pane will show events and the Job Inspector will show an **Events collected** count.

However, if previewing returns no events and throws no error, first check your **Filter** expression by previewing without it by simplifying the **Filter** expression to `true`. Then check the Job Inspector: If the **Total size** is greater than `0`, and the **Received size** is `NA` or `0`, make sure you have list and read permissions on the object store.

If you're still not seeing events, try setting your S3 Collector's **Path expression** to a `/`.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These "meta" fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object's nested fields contain metadata about each collection job.

15.1.9. Script

Cribl Stream supports flexible data collection configured by your custom scripts.

In Cribl.Cloud, the Script Collector is only available on hybrid, customer-managed Worker Nodes.

How the Collector Pulls Data

When you run a Script Collector in Discovery mode, the first available Worker returns one line of data per discovered item. Each item (line) turns into a Collection task on the Leader Node.

In Full Run mode, the Leader passes the item to collect into the script in the `$CRIBL_COLLECT_ARG` variable, and spreads collection across all available Workers.

If you run the Script Collector with a given Time Range, the `earliest` and `latest` values will be available to the script via the environment variables `EARLIEST` and `LATEST`, respectively.

Script Caveats

Scripts will allow you to execute almost anything on the system where Cribl Stream is running. Make sure you understand the impact of what you're executing before you do so! These scripts run as the user running Cribl Stream, so if you are running it as root, these commands will run with root user permissions.

Scripts must finish on their own. The Collector will not stop a script once its task has started. You must ensure scripts do not use infinite loops.

Configuring a Script Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Script** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **Script** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. E.g., `sh2GetStuff`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Discover script: Script to discover which objects/files to collect. This script should output one task per line in `stdout`. Discovery is especially important in a distributed deployment, where the Leader must track all tasks, and must guarantee that each is run by a single Worker. See Examples below.

Collect script: Script to perform data collections. Pass in tasks from the Discover script as `$CRIBL_COLLECT_ARG`. Should output results to `stdout`.

Optional Settings

Shell: Shell in which to execute scripts. Defaults to `/bin/bash`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

How the Collector Pulls Data

In the Discover phase, the first available Worker returns one line of data per item discovered. Each item line turns into a Collect task on the Leader Node. In the Collect phase, the items to collect are passed into the script as the variable `$CRIBL_COLLECT_ARG`, and are spread across all available Workers.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These "meta" fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object's nested fields contain metadata about each collection job.

Examples

Telemetry Collector

You could define this Collector to check for Cribl Stream telemetry errors, which could cause license validation to fail, eventually (after a delay) blocking data input.

Collector type: `Script`

Discover script: `ls $CRIBL_HOME/log/cribl*`

Collect script: `grep 'Failed to send anonymized telemetry metadata' $CRIBL_COLLECT_ARG`

S3 Collector

In this example, the Discover script retrieves file names from a specified Amazon S3 bucket, and then writes them (one per line) to the standard output. The Collect script processes each line as its `$CRIBL_COLLECT_ARG`, and uses `zcat` to decompress the buckets' data.

Collector type: `Script`

Discover script:
`aws s3api list-objects --bucket <bucket-name> --prefix <subfolder>/ --query 'Contents[].Key' --output text`

Collect script: `aws s3 cp s3://<bucket-name>/$CRIBL_COLLECT_ARG - | zcat -f`

Simple Collector

This example essentially spoofs the Discover script with an `echo` command, which simply announces what the Collect script (itself simple) will do.

Collector type: `Script`

Discover script: `echo "speedtest"`

Collect script: `speedtest --json`

15.1.10. Splunk Search

Cribl Stream supports collecting search results from Splunk queries. The queries can be both simple and complex, as well as real-time searches. This page covers how to configure the Collector.

Configuring a Splunk Search Collector

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, select **Data** > **Sources**, then select **Collectors** > **Splunk Search** from the **Manage Sources** page's tiles or left nav. Click **Add Collector** to open the **Splunk Search** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links at left to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

Collector Settings

The Collector Settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. E.g., `splunk2search`. If you clone this Collector, Cribl Stream will add `-CLONE` to the original **Collector ID**.

Search endpoint: Rest API used to conduct a search. Defaults to `services/search/jobs/export`.

Output mode: Format of the returned output. Defaults to JSON format.To parse the returned JSON, add the Cribl event breaker which parses newline delimited events in the Event Breakers tab.

Events returned from Splunk search can also be returned in the more compact CSV format. To use CSV format, set the **Output mode** to CSV and specify the CSV event breaker in the Event Breakers tab.

Search

In the **Search** dropdown, type your query parameters:

Search: Enter the Splunk query. For example: `index=myAppLogs level=error channel=myApp OR | mstats avg(myStat) as myStat WHERE index=myStatsIndex`.

Search head: Enter the search head base URL. The default is `https://localhost:8089`.

Earliest: You can enter the earliest time boundary for the search. This maybe be an exact or relative time. For example: `2022-01-14T12:00:00Z` or `-16m@m`.

Latest: You can enter the latest time boundary for the search. This maybe be an exact or relative time. For example: `2022-01-14T12:00:00Z` or `-16m@m`.

Authentication

In the **Authentication** drop-down, select one of these options:

- **None**: Don't use authentication. Compatible with REST servers like AWS, where you embed a secret directly in the request URL.
- **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.
- **Bearer Token**: Provide the `token` value configured and generated in Splunk.
- **Bearer Token (text secret)**: Provide the Bearer Token referenced by a secret. Select a stored text secret in the resulting **Token (text secret)** drop-down, or click **Create** to configure a new secret.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Optional Settings

Extra parameters: Optional HTTP request parameters to append to the request URL. These refine or narrow the request. Click **Add Parameter** to add parameters as key-value pairs:

- **Name**: Field name.
- **Value**: JavaScript expression to compute the field's value (can be a constant).

Extra headers: Click **Add Header** to (optionally) add collection request headers as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value (can be a constant).

In the **Request Timeout (secs)** field, you can set a maximum time period (in seconds) for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `0`, which disables timeout metering.

> When running a real-time search you must update the **Request Timeout Parameter** to avoid having the collector stuck in a forever running state. Updating the **Request Timeout Parameter** stops the search after the allocated period of time.

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.

Disable time filter: Toggle to `Yes` if your Run or Schedule configuration specifies a date range and no events are being collected. This will disable the Collector's event time filtering to prevent timestamp conflicts.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (e.g., self-signed certificates). Defaults to `No`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Result Settings

The Result Settings determine how Cribl Stream transforms and routes the collected data.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Result Routing

Send to Routes: If set to `Yes` (the default), Cribl Stream will send events to normal routing and event processing. Toggle to `No` to select a specific Pipeline/Destination combination. The `No` setting exposes these two additional fields:

- **Pipeline**: Select a Pipeline to process results.
- **Destination**: Select a Destination to receive results.

The default `Yes` setting instead exposes this field:

- **Pre-processing Pipeline**: Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

You might disable **Send to Routes** when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. One use case might be a Splunk Search Collector that gathers a known, simple type of data. This approach keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Pre-processing Pipeline: Pipeline to process results before sending to Routes. Optional, and available only when **Send to Routes** is toggled to `Yes`.

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. (Example: `42 MB`.) Default value of `0` indicates no throttling.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

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.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to `4h`.

Remove Discover fields : List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle to `Yes` to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Internal Fields

Cribl Stream uses a set of internal fields to assist in data handling. These “meta” fields are **not** part of an event, but they are accessible, and you can use them in Functions to make processing decisions.

Relevant fields for this Collector:

- `__collectible` – This object’s nested fields contain metadata about each collection job.

How Cribl Stream Pulls Data

This Collector will gather data from the specified **Search head** URL. If you enable scheduled collection, searches will repeat on the interval specified in the **Schedule** modal’s **Cron schedule** field.

A single Worker executes each collection job. If the Leader goes down, search jobs in progress will complete, but future scheduled searches will not run until the Leader relaunches.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a “stuck”, “orphaned,” or “zombie” job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream’s logs.

To keep stuck jobs from running excessively long:

- First, try setting **Optional Settings** > **Request Timeout (secs)** – whose default of `0` means “wait forever” – to a desired maximum duration.
- If adjusting **Timeout (secs)** does not fix the problem, try the global setting **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration. You’ll find **Job Timeout** among the task manifest and buffering limits.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

15.1.11. Manage Collector State In Database And REST API Collectors

Manage State in Database Collector

For the Database Collector, **state** identifies the last row that the Collector has read from the database, as expressed by that row's value in the tracking column. You can view, edit, refresh, or delete that state – and that is what we mean by managing state. Testing or debugging a Collector is the main reason to do this.

Manage State in REST Collector

The REST Collector tracks **state** according to configuration of the expressions **State update** and **State merge**. With state tracking enabled, these expressions default to track state by the greatest event `_time` value.

You can enable state tracking in the REST Collector in **State Tracking** when you schedule or perform a full run.

Manage State

The **Manage State** button is enabled whenever your Collector's state is not empty, and is available from two locations:

- In the **Database** list view of your Database Collectors, at the end of each Collector's row. Click **Manage**, **Data**, **Manage Sources**, **Collectors**, **Database** to reach this UI.
- In the configuration modal for each Collector.

To view state, click the button. This opens the **Manage State** JSON editor, since state is recorded as a JSON object. For example, if you have chosen the `id` column in your database as your tracking column, and your Collector has just read 5 rows, the state object will look like this:

```
{
  "id": {
     "value": 5
  }
}
```

Suppose a particular job did not read the data you expected it to. Viewing state is the most basic troubleshooting step in this kind of situation.

If you know that state was updated because the Collector has run, but the state shown in the editor does not reflect the change you expected, click **Refresh State**.

If you want to start again with no state, click **Delete State**. To learn how and why to use this operation, see Deleting Collector State, below.

Edit Collector State

In the **Manage State** JSON editor, you can change state by editing the JSON, and then click **Save State** to persist the change.

To continue our example, if your Collector has read 1000 rows but you want it to repeat the last 500 rows on the next scheduled run, you would change the state from `1000` to `500`.

Delete Collector State

The **Delete State** and **Delete Collector** buttons are independent of each other. Deleting the Collector does **not** delete its saved state information. Cribl Stream decouples these two operations to provide benefits that we'll explore in this section.

To begin with, note that Collector state is stored on disk, not in Git. Each Collector's state is stored separately.

Secondly, the uses of state-deleting differ depending on whether your Database Collector is on a single-instance deployment of Cribl Stream, or a distributed deployment.

Delete Collector State in a Single-Instance Deployment

When you first configure a Database Collector, you might need to experiment with SQL queries until you arrive at one that returns the desired results. In doing this, you might need the queries you're testing to always start from the same place in the database. If you're using a tracking column, this won't work, because each query will move you further along in the database. The solution is to delete state between iterations of the query you're building.

Delete Collector State in a Distributed Deployment

Just like in single-instance deployments, in distributed deployments it can be useful to delete Collector state when you're testing SQL queries for a Collector with state tracking enabled.

Beyond that, with distributed deployments it’s important to know that Cribl Stream decouples the deleting of Collectors themselves from the deleting of their saved state, and to see the benefits of that decoupling.

Suppose you have been running a Collector with state tracking enabled, and it’s working its way through a database. At some point you decide you’re not happy with that Collector and you click the **Delete Collector** button. What happens next? There are at least three possibilities:

- If you’re sure you want to delete the Collector, you should Commit and Deploy. Once you do that, the Leader pushes the change out to the Worker Nodes, and the Collector is gone (although you can recover it by rolling back to a previous commit in Git).
- If you realize that you made a mistake, and want to keep the Collector after all, then you should **not** Commit and Deploy. Instead, click the **Undo All Changes** button in the **Git Changes** modal – this un-deletes the Collector. And, since the Collector’s state remains on disk, the Collector will pick up where it left off.
- If you deleted the Collector because you want to create another Collector with the same ID and the same tracking column, but a different configuration, you should Commit and Deploy, then create your new Collector. Because the old Collector’s state remains on disk, the new Collector with the same ID picks up the saved state. When you begin using the new Collector, it will pick up where the old Collector left off. On the other hand, if you prefer the new Collector to start from zero, just click the **Delete State** button before you start using it.

In short, Cribl Stream’s decoupling of Collector deletion from state deletion gives you control, such that you can avert messy situations where your Collector cannot find the state it needs.

15.1.12. Scheduling And Running

Once you've configured a Collector, you can either run it immediately ("ad hoc") to collect data, or schedule it to run on a recurring interval. Scheduling requires some extra configuration upfront, so we cover this option first.

For **ad hoc** collection, you can configure whether a job interrupted by an unintended Cribl Stream shutdown will automatically resume upon Cribl Stream restart.

But regardless of this configuration, if you **explicitly** restart or stop Cribl Stream, this will cancel any currently running jobs. This applies to executing the `./cribl restart` or `./cribl stop` CLI commands, as well as to selecting the UI's **Settings** > **Global Settings** > **Controls** > **Restart** option.

A **scheduled** job interrupted by a shutdown (whether explicit or unintended) will **not** resume upon restart.

Schedule Configuration

Click **Schedule** beside a configured Collector to display the **Schedule configuration** modal. This provides the following controls.

Enabled: Slide to `Yes` to enable this collection schedule.

The scheduled job will keep running on this schedule forever, unless you toggle **Enabled** back to `Off`. The `Off` setting preserves the schedule's configuration, but prevents its execution.

Cron schedule: A cron schedule on which to run this job.

- The **Estimated schedule** below this field shows the next few collection runs, as examples of the cron interval you've scheduled.

Max concurrent runs: Sets the maximum number of instances of this scheduled job that Cribl Stream will simultaneously run.

Skippable: Skippable jobs can be delayed up to their next run time if the system is hitting concurrency limits. Defaults to `Yes`. Toggling this to `No` displays this additional option:

- **Resume missed runs**: Defaults to `No`. When toggled to `Yes`, if Cribl Stream's Leader (or single instance) restarts, it will run all missed jobs according to their original schedules.

Skippable Jobs and Concurrency Limits

If set to `Yes`, the **Skippable** option obeys these concurrency limits configured separately in **Settings** > **Global Settings** > **General Settings** > **Job Limits**:

- **Concurrent Job Limit**
- **Concurrent Scheduled Job Limit**

See Job Limits for details on these and other limits that you can set in global **Settings**.

When the above limits delay a Skippable job:

- The Skippable job will be granted slightly higher priority than non-Skippable jobs.
- If the job receives resources to run before its next scheduled run, Cribl Stream will run the delayed job, then snap back to the original cron schedule.
- If resources do **not** free up before the next scheduled run: Cribl Stream will **skip** the delayed run, and snap back to the original cron schedule.

Set **Skippable** to `No` if you absolutely must have all your data, for compliance or other reasons. In this case, Cribl Stream will build up a backlog of jobs to run.

You can think of **Skippable**: `No` as behaving more like the TCP protocol, with **Skippable**: `Yes` behaving more like UDP.

All collection jobs are constrained by the following options in **Settings** > **Global Settings** > **General Settings** > **Job Limits**:

- **Concurrent Task Limit**
- **Max Task Usage Percentage**

Run Configuration and Shared Settings

Most of the remaining fields and options below are shared with the **Run configuration** modal, which you can open by clicking **Run** beside a configured Collector.

Mode

Depending on your requirements, you can schedule or run a collector in these modes:

- Preview – Default for Run, but not offered for scheduled jobs.
- Discovery – Default for scheduled jobs.
- Full Run.

Preview

In the Preview mode, a collection job will return only **a sample subset** of matching results (e.g., 100 events). This is very useful in cases when users need a data sample to:

- Ensure that the correct data comes in.
- Iterate on Filter expressions.
- Capture a sample to iterate on Pipelines.

Schedule configuration omits the Preview option, because Preview is designed for immediate analysis and decision making. To configure a scheduled job with high confidence, you can first manually run Preview jobs with the same Collector, to verify that you're collecting the data you expect.

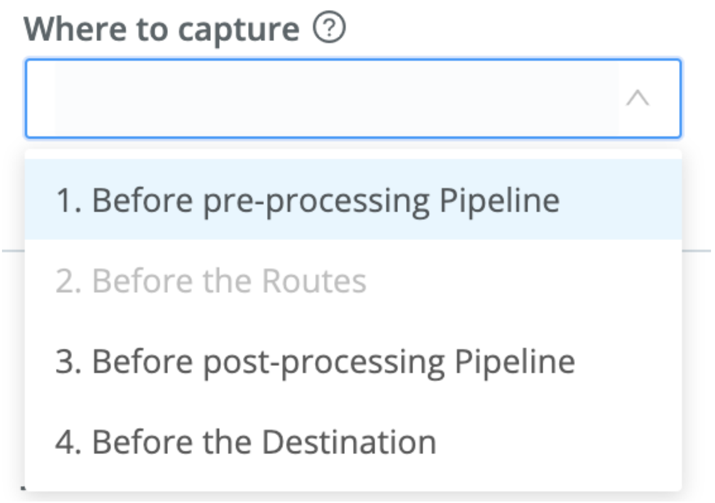
Preview Settings

In Preview mode, you can optionally configure these options:

Capture time (sec): Maximum time interval (in seconds) to collect data.

Capture up to N events: Maximum number of events to capture.

Where to capture: Select one of the options shown below. (Note that option `2. Before the Routes` is disabled.) If not specified, this will default to `1. Before pre-processing Pipeline`.


Where to capture
1. Before pre-processing Pipeline
2. Before the Routes
3. Before post-processing Pipeline
4. Before the Destination

Preview capture options

Discovery

In Discovery mode, a collection job will return only **the list of objects/files** to be collected, but none of the data. This mode is typically used to ensure that the Filter expression and time range are correct before a Full Run job collects unintended data.

Send to Routes

In Discovery mode, this toggle enables you to send discovery results to Cribl Stream Routes. Defaults to `No`.

> This setting overrides the Collector configuration's **Result Routing** > **Send to Routes** setting.

Full Run

In Full Run mode, the collection job is fully executed by Worker Nodes, and will return all data matching the Run configuration. Optionally, for the Database Collector, you can also configure state tracking.

Time Range

Set an **Absolute** or **Relative** time range for data collection. The **Relative** option is the default, and is particularly useful for configuring scheduled jobs.

> You set dates and times here based on your browser's time zone, but Cribl Stream's backend uses UTC. Set the **Range Timezone** drop-down to your offset from UTC.

Absolute

Select the **Absolute** button to set fixed collection boundaries in your browser's local time. Next, use the **Earliest** and **Latest** controls to set the start date/time and end date/time.

Relative

Select the **Relative** button to set collection boundaries relative to your browser's current local time. Next, use the **Earliest** and **Latest** fields to set start and end times like these:

- **Earliest** example values: `-1h`, `-42m`, `-42m@h`
- **Latest** example values: `now`, `-20m`, `+42m@h`

By default, Cribl Stream uses **Earliest** and **Latest** as time-based tokens, which it evaluates in two ways:

- By comparison to time values specified in the directory path.
- By comparison to the `_time` field.

The Google Cloud Storage, REST/API, and Splunk Search Collectors provide a **Disable time filter** option in their **Collector Settings**. When set to `Yes`, scheduling or running these Collectors against a date range will cause Cribl Stream to ignore event time–based filtering entirely.

Relative Time Syntax

For Relative times, the **Earliest** and **Latest** controls accept the following syntax:

```
[+|-]<time_integer><time_unit>@<snap-to_time_unit>
```

To break down this syntax:

Syntax Element	Values Supported
Offset	Specify: `-` for times in the past, `+` for times in the future, or omit with `now`.
<time_integer>	Specify any integer, or omit with `now`.
<time_unit>	Specify the `now` constant, or one of the following abbreviations: `s[econds]`, `m[inutes]`, `h[ours]`, `d[ays]`, `w[eeks]`, `mon[ths]`, `q[uarters]`, `y[ears]`.
@<snap-to_time_unit>	Optionally, you can append the `@` modifier, followed by any of the above `<time_unit>`s, to round down to the nearest instance of that unit. (See the next section for details.)

Cribl Stream validates relative time values using these rules:

- **Earliest** must not be later than **Latest**.
- Values without units get interpreted as seconds. (E.g., `-1` = `-1s`.)

Snap-to-Time Syntax

The `@` snap modifier always rounds **down** (backwards) from any specified time. This is true even in relative time expressions with `+` (future) offsets. For example:

- `@d` snaps back to the beginning of today, 12:00 AM (midnight).

- `+128m@h` looks forward 128 minutes, then snaps back to the nearest round hour. (If you specified this in the `Latest` field, and ran the Collector at 4:20 PM, collection would end at 6:00 PM. The expression would look forward to 6:28 PM, but snap back to 6:00 PM.)

Other options:

- `@w` or `@w7` to snap back to the beginning of the week – defined here as the preceding Sunday.
- To snap back to other days of a week, use `w1` (Monday) through `w6` (Saturday).
- `@mon` to snap back to the 1st of a month.
- `@q` to snap back to the beginning of the most recent quarter – Jan. 1, Apr. 1, Jul. 1, or Oct. 1.
- `@y` to snap back to Jan. 1.

Filter

This is a JavaScript filter expression that is evaluated against token values in the provided collector path (see below), and against the events being collected. The **Filter** value defaults to `true`, which matches all data, but this value can be customized almost arbitrarily.

For example, if a File System or S3 collector is run with this Filter:

```
host=='myHost' && source.endsWith('.log') || source.endsWith('.txt')
```

...then only files/objects with `.log` or `.txt` extensions will be fetched. And, from those, only those events with host field `myHost` will be collected.

At the **Filter** field's right edge are a Copy button, an Expand button to open a validation modal, and a History button. For more extensive options, see Tokens for Filtering below.

Advanced Settings

Log Level: Level at which to set task logging. More-verbose levels are useful for troubleshooting jobs and tasks, but use them sparingly.

Lower task bundle size: Limits the bundle size for small tasks. E.g., bundle five 200KB files into one 1MB task bundle. Defaults to `1MB`.

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. Defaults to `10MB`.

Reschedule tasks: Whether to automatically reschedule tasks that failed with non-fatal errors. Defaults to `Yes`; does not apply to fatal errors.

Max task reschedule: Maximum number of times a task can be rescheduled. Defaults to `1`.

Job timeout: Maximum time this job will be allowed to run. Units are seconds, if not specified. Sample values: `30`, `45s`, or `15m`. Minimum granularity is 10 seconds, so a `45s` value would round up to a 50-second timeout. Defaults to `0`, meaning unlimited time (no timeout).

State Tracking

Enabled: Toggle to `Yes` if you want to enable state tracking when running scheduled or full run jobs.

Subsequent options depend on the type of Collector you're working with.

For Database Collectors, see Working with a Tracking Column to learn how to craft a Collector query for state tracking.

For REST Collectors, see Working with State Tracking to learn how enable and use state in your collection configuration.

State tracking is available only for Database and REST Collectors.

Cribl recommends that when you enable state tracking on a Collector, you configure the schedule to allow only one job to run at a time.

This prevents one run from starting before another has finished, which avoids the collection of duplicate records and/or incorrect state tracking.

Tokens for Filtering

Let's look at the options for path-based (basic) and time-based token filtering.

Basic Tokens

In collectors with paths, such as Filesystem or S3, Cribl Stream supports path filtering via token notation. Basic tokens' syntax follows that of JS template literals: `${<token_name>}` – where `token_name` is the field (name) of interest.

For example, if the path was set to `/var/log/${hostname}/${sourcetype}/`, you could use a Filter such as `hostname=='myHost' && sourcetype=='mySourcetype'` to collect data only from the `/var/log/myHost/mySourcetype/` subdirectory.

Time-based Tokens

In paths with time partitions, Cribl Stream supports further filtering via time-based tokens. This has a direct effect with earliest and latest boundaries. When a job runs against a path with time partitions, the job traverses a minimal superset of the required directories to satisfy the time range, before subsequent event `_time` filtering.

About Partitions and Tokens

Cribl Stream processes time-based tokens as follows:

- For each path, time partitions must be notated in descending order. So Year/Month/Day order is supported, but Day/Month/Year is not.
- Paths may contain more than one partition. E.g., `/my/path/2020-04/20/`.
- In a given path, each time component can be used only once.
  So `/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...` is a valid expression format, but `/my/path/${_time:%Y}/${_time:%m}/${host}/${_time:%Y}/...` (with a repeated `Y`) is not supported.
- For each path, all extracted dates/times are considered in UTC.

The following strptime format components are allowed:

- `'Y'`, `'y'`, for years
- `'m'`, `'B'`, `'b'`, `'e'`, for months
- `'d'`, `'j'`, for days
- `'H'`, `'I'`, for hours
- `'M'`, for minutes
- `'S'`, for seconds

Token Syntax

Time-based token syntax follows that of a slightly modified JS template literal:
`${_time: <some_strptime_format_component>}`. Examples:

Filter	Matches
`/my/path/${_time:%Y}/${_time:%m}/${_time:%d}/...`	`/my/path/2020/04/`
`/my/path/${_time:year=%Y}/${_time:month=%m}/${_time:day=%d}/...`	`/my/path/year=202`
`/my/path/${_time:%Y-%m-%d}/...`	`/my/path/2020-05-`

Autoscaling with Collection Jobs

Cribl Stream supports autoscaling for streaming data, but not for collection jobs.

If you run a collection job using Workers in an autoscaling group (such as an AWS instance), you must adequately size your Worker Group or Worker Groups and provision the Worker Nodes before you run or schedule a collection job. Otherwise, a race condition may occur between task scheduling and configuration download from the Leader.

15.1.13. Job Limits

You can configure global limits that optimize the execution of all Collectors and scheduled jobs (including Cribl Stream system tasks). To access and set these limits, follow the steps for your deployment type:

- Single-instance deployments: **Settings** > **System** > **General Settings** > **Limits** > **Jobs**.
  These limits apply to all Worker Processes.
- Distributed deployments: From the Leader, [select a Worker Group] > **Group Settings** > **System** > **General Settings** > **Limits** > **Jobs**.
  These limits are applied at the Worker Group level (except where noted), and trickle down to individual Worker Processes in the Group.

Limits Available

The following controls are available at the UI locations listed above.

Job Limits

Disable Jobs/Tasks: Set to `No` by default in Stream Groups. When enabled, Nodes running 4.5.0 and newer will not poll the Leader for jobs/tasks, and Nodes will not use the settings below (even if they're set).

Nodes running 4.4.4 and older still upgrade via jobs and will honor the Jobs settings, even with **Disable jobs/tasks** toggled to `Yes`.

Concurrent Job Limit: The total number of jobs that can run concurrently. Defaults to `10`.

Concurrent System Job Limit: The total number of **system** jobs that can run concurrently. Defaults to `10`. Minimum `1`.

Concurrent Scheduled Job Limit: The total number of **scheduled** jobs that can run concurrently. This limit is set as an offset relative to the **Concurrent Job Limit**. Defaults to `-2`.

Skipped jobs indicate that a Group's **Concurrent Job Limit** has been reached or exceeded. Increase this limit to reduce the number of skippable jobs. For resource-intensive jobs, this might require deploying more Worker Nodes.

Task Limits

Concurrent Task Limit: The total number of tasks that a Worker Process can run concurrently. Defaults to `2`. Minimum `1`.

Concurrent System Task Limit: The number of system tasks that a Worker Process can run concurrently. Defaults to `1`. Minimum `1`.

Max Task Usage Percentage: Value, between `0` and `1`, representing the percentage of total tasks on a Worker Process that any single job may consume. Defaults to `0.5` (i.e., 50%).

Task Poll Timeout: The number of milliseconds that a Worker's task handler will wait to receive a task, before retrying a request for a task. Defaults to `60000` (i.e., 60 seconds). Minimum `10000` (10 seconds).x

Completion Limits

Artifact Reaper Period: Interval on which Cribl Stream attempts to reap jobs' stale disk artifacts. Defaults to `30m`.

Finished Job Artifacts Limit: Maximum number of finished job artifacts to keep on disk. Defaults to `100`. Minimum `0`.

Finished Task Artifacts Limit: Maximum number of finished task artifacts to keep on disk, per job, on each Worker Node. Defaults to `500`. Minimum `0`.

Task Manifest and Buffering Limits

Manifest Flush Period: The rate (in milliseconds) at which a job's task manifest should be refreshed. Defaults to `100` ms. Minimum `100`, maximum `10000`.

Manifest Max Buffer Size: The maximum number of tasks that the task manifest can hold in memory before flushing to disk. Defaults to `1000`. Minimum `100`, maximum `10000`.

Manifest Reader Buffer Size: The number of bytes that the task manifest reader should pull from disk. Defaults to `4kb`.

Job Dispatching: The method by which tasks are assigned to Worker Processes. Defaults to `Least In-Flight Tasks`, to optimize available capacity. `Round Robin` is also available.

Job Timeout: Maximum time a job is allowed to run. Defaults to `0`, for unlimited time. Units are seconds if not specified. Sample entries: `30`, `45s`, `15m`.

Task Heartbeat Period: The heartbeat period (in seconds) for tasks to report back to the Leader/API. Defaults to `60` seconds. Minimum `60`.

15.2. Amazon

15.2.1. Amazon Kinesis Firehose

Cribl Stream 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 Stream to Receive Data over HTTP(S) from Amazon Kinesis Firehose

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`
- `__raw`
- `__srcIpPort` – See details below.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the 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 Stream can buffer, causing Cribl Stream to send HTTP 500 error responses to Kinesis Firehose.

15.2.2. Amazon Kinesis Streams

Cribl Stream supports receiving data records from Amazon Kinesis Streams.

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **No**

Configuring Cribl Stream to Receive Data from Kinesis Streams

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Amazon** > **Kinesis**. 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 Kinesis Stream Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Stream name: Kinesis stream name (not ARN) to read data from.

Region: Region where the Kinesis stream is located. Required.

Optional Settings

Shard iterator start: Location at which to start reading a shard for the first time. Defaults to `Earliest Record`.

Record data format: Format of data inside the Kinesis Stream records. Gzip compression is automatically detected. Options include:

- Cribl (the default): Use this option if Cribl Stream wrote data to Kinesis in this format. This is a type of NDJSON.

- Newline JSON: Use if the records contain newline-delimited JSON (NDJSON) events – e.g., Kubernetes logs ingested through Kinesis. This is a good choice if you don't know the records' format.
- CloudWatch Logs: Use if you've configured CloudWatch to send logs to Kinesis.
- **Event per line**: NDJSON can use this format when it fails to parse lines as valid JSON.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down 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 Stream 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.
- **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.

Auto Authentication

When using an IAM role to authenticate with Kinesis Streams, the IAM policy statements must include the following Actions:

- `kinesis:GetRecords`
- `kinesis:GetShardIterator`
- `kinesis:ListShards`

For details, see AWS' Actions, resources, and condition keys for Amazon Kinesis Data Streams documentation.

Manual Authentication

The **Manual** option exposes these 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.

Secret Authentication

The **Secret** option exposes this additional field:

Secret key pair: Use the drop-down to select a secret key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS. Follow the **Create** link if you need to configure a key pair.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for Kinesis Streams: Whether to use Assume Role credentials to access Kinesis Streams. Defaults to `No`.

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

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

Shard selection expression: A JavaScript expression to be called with each `shardId` for the stream. The shard will be processed if the expression evaluates to a truthy value. Defaults to `true`.

Service Period: Time interval (in minutes) between consecutive service calls. Defaults to `1` minute.

Endpoint: Kinesis stream service endpoint. If empty, the endpoint will be automatically constructed from the region.

Signature version: Signature version to use for signing Kinesis Stream requests. Defaults to `v4`.

Verify KPL checksums: Enable this setting to verify Kinesis Producer Library (KPL) event checksums. This setting is not available when Cribl Stream is deployed in FIPS mode.

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`.

Avoid duplicate records: If toggled to `Yes`, this Source will always start streaming at the next available record in the sequence. (This can cause data loss after a Worker Node's unexpected shutdown or restart.) With the default `No`, the Source will reread the last two batches of events at startup. (This prevents data loss, but can ingest duplicate events.)

Records limit per call: In v.4.0.4 and later, sets the maximum number of records ingested, per shard, in each getRecords call. You can acclerate data collection by increasing the default/minimum `5000` limit, to as high as `10000` records.

Total records limit: In v.4.0.4 and later, sets the maximum number of records, across all shards, to pull down at once per Worker Process. Default/minimum value is `20000` records. Beware of very high values that could excessively consume Workers' memory.

Shard Load Balancing: The load-balancing algorithm to use for spreading out shards across Workers and Worker Processes. Options include:

- Consistent Hashing (default)
- Round Robin

For the Round Robin option, both the Leader and the Worker Groups must be on Cribl Stream 4.1 or later.

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:

- `__checksum`
- `__encryptionType`
- `__final`
- `__inputID`
- `__isKPL`
- `__partitionKey`
- `__sequenceNumber`
- `_raw`

How Cribl Stream Pulls Data

Worker Processes get a list of available shards from Kinesis, and contact the Leader Node to fetch the latest sequence numbers. Based on the sequence number's value, the Worker either resumes the shard reading from where Cribl Stream previously left off, or starts reading from the beginning.

The Kinesis Streams Source stores shard state on disk, so that it can pick up where it left off across restarts. The state file is located in Cribl Stream's `state/` subdirectory; the path format looks like this:

```
.../state/kvstore/<groupId>/input_kinesis_<inputId>_<streamName>/state.json
```

For example:

```
state/kvstore/default/input_kinesis_kinesisIn_just-a-test/state.json
```

Worker Processes become Kinesis Consumers, and fetch the records for the assigned shards. Every 5 minutes, each Worker Process forwards to the Leader Node the latest sequence numbers for the shards that Worker Process is responsible for. The Leader Node persists the `shardId` > `sequenceNumber` mapping to disk.

15.2.3. Amazon S3

Cribl Stream supports receiving data from Amazon S3 buckets, using event notifications through Amazon Simple Queue Service (SQS).

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **YES**

Cribl Stream running on Linux (only) can use this Source to read Parquet files, identified by a `.parquet`, `.parq`, or `.pqt` filename extension.

See our Amazon S3 Better Practices and Using S3 Storage and Replay guides.

S3 Setup Strategy

The source S3 bucket must be configured to send `s3:ObjectCreated:*` events to an SQS queue, either directly (easiest) or via SNS (Amazon Simple Notification Service). See the event notification configuration guidelines below.

SQS messages will be deleted after they're read, unless an error occurs, in which case Cribl Stream will retry. This means that although Cribl Stream will ignore files not matching the **Filename Filter**, their SQS events/notifications will still be read, and then deleted from the queue (along with those from files that match).

These ignored files will no longer be available to other S3 Sources targeting the same SQS queue. If you still need to process these files, we suggest one of these alternatives:

- Using a different, dedicated SQS queue. (Preferred and recommended.)
- Applying a broad filter on a single Source, and then using pre-processing Pipelines an/or Route filters for further processing.

Compression

Cribl Stream can ingest compressed S3 files if they meet all the following conditions:

- Compressed with the `x-gzip` MIME type.
- End with the `.gz` extension.
- Can be uncompressed using the `zlib.gunzip` algorithm.

Storage Class Compatibility

Cribl Stream does **not** support data preview, collection, or replay from S3 Glacier or S3 Deep Glacier storage classes, whose stated retrieval lags (variously minutes to 48 hours) cannot guarantee data availability when the Collector needs it.

Cribl Stream **does** support data preview, collection, and replay from S3 Glacier Instant Retrieval when you're using the S3 Intelligent-Tiering storage class.

Configuring Cribl Stream to Receive Data from Amazon S3

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Amazon** > **S3**. 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 S3 Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Queue: The name, URL, or Amazon Resource Name (ARN) of the SQS queue to read events from. When specifying a non-AWS URL, you must use the format: `{url}/<queueName>`. (For example: `https://host:port/<queueName>`.) This value must be a JavaScript expression (which can evaluate to a constant), enclosed in single quotes, double quotes, or backticks.

Optional Settings

Filename filter: Regex matching file names to download and process. Defaults to `.*`, to match all characters. This regex will be evaluated against the S3 key's full path.

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.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down 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 Stream 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, such as 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 a secret key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS. Follow the **Create** link if you need to configure a key pair.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for S3: Whether to use Assume Role credentials to access S3. Defaults to `Yes`.

Enable for SQS: Whether to use Assume Role credentials when accessing SQS. Defaults to `No`.

AWS account ID: SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account.

AssumeRole ARN: Enter the 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

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

This section defines event breaking rulesets that will be applied, in order.

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

Checkpoint Settings

Enable checkpointing: When toggled to `No` (the default) Cribl Stream will always start processing a file from the beginning. Toggle to `Yes` if, after an error or restart occurs, you want Cribl Stream to resume processing a given file from the point where it was interrupted, as explained below. When turned on, the following additional setting becomes available:

- **Retries:** The number of times you want Cribl Stream to retry processing a file after an error or restart occurs. If **Skip file on error** is enabled, this setting is ignored.

Other Settings

Endpoint: S3 service endpoint. If empty, defaults to AWS's region-specific endpoint. Otherwise, used to point to an S3-compatible endpoint. 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 Stream will automatically add it to the URL path. For details, see AWS' Path-Style Requests topic.

Signature version: Signature version to use for signing SQS requests. Defaults to `v4`.

Max messages: The maximum number of messages that SQS should return in a poll request. Amazon SQS never returns more messages than this value. (However, fewer messages might be returned.) Acceptable values: 1 to 10. Defaults to `1`.

Visibility timeout seconds: The duration (in seconds) that the received messages are hidden from subsequent retrieve requests, after being retrieved by a ReceiveMessage request. Defaults to `600`.

> Cribl Stream will automatically extend this timeout until the initial request's files have been processed – notably, in the case of large files that require additional processing time.

Num receivers: The number of receiver processes to run. The higher the number, the better the throughput, at the expense of CPU overhead. Defaults to `1`.

Poll timeout (secs): How long to wait for events before polling again. Minimum `1` second; default `10`; maximum `20`. Short durations increase the number - and thus the cost – of requests sent to AWS. (The UI will show a warning for intervals shorter than `5` seconds.) Long durations increase the time the Source takes to react to configuration changes and system restarts.

Socket timeout: Socket inactivity timeout (in seconds). Increase this value if retrievals time out during backpressure. Defaults to `300` seconds.

Max Parquet chunk size (MB): Maximum size for each Parquet chunk. Defaults to `5` MB. Valid range is `1` to `100` MB. Cribl Stream stores chunks in the location specified by the `CRIBL_TMP_DIR` environment variable. It removes the chunks immediately after reading them. See Environment Variables.

Parquet chunk download timeout (seconds): The maximum time to wait for a Parquet file's chunk to be downloaded. If a required chunk cannot not be downloaded within this time limit, processing will end. Defaults to `600` seconds. Valid range is `1` second to `3600` seconds (1 hour).

Skip file on error: Toggle to `Yes` to skip files that trigger a processing error (for example, corrupted files). Defaults to `No`, which enables retries after a processing error.

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (for example, self-signed certificates). Defaults to `Yes`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Checkpointing

When Workers restart, checkpointing can prevent the Source from losing data or collecting duplicate events. Among the situations where this works well:

- When Cribl Stream pushes a new configuration: the Leader pushes the configuration to the Workers, and restarts them.
- When a Worker has gotten into a bad state, and you want to restart it.

Checkpointing allows Cribl Stream to resume processing a file at the point where an error or restart occurred, minimizing the number of duplicate events sent downstream in the event of an interruption. To accomplish this, Cribl Stream keeps track of how many events have been processed for a given file. Resuming processing after an interruption, the Source skips the first *N* events it has seen and starts processing from event *N*+1.

The Source manages checkpoint information in memory, but writes it out to disk periodically (once per second, and on Worker shutdown) so that the information can survive a Worker restart. Cribl Stream keeps checkpoint information at the Worker Process level and does not share it between Worker Processes or Worker Nodes.

Checkpointing is a feature to keep in mind when you do disaster recovery (DR) planning.

Caveats to using checkpointing include: Whenever Worker Node state is destroyed (that is, when a Worker ceases to exist), checkpoint information will be lost. In Cribl.Cloud deployments, this can happen during upgrades. Also, because checkpoints are not written to disk on **every** event, the checkpoints written to disk will lag behind those that the S3 Source manages in memory. If the Worker Process stops abruptly without a clean shutdown, checkpoints not yet written to disk will be lost. When processing resumes, the Source will start from just after the last checkpoint written to disk, producing duplicate events corresponding to the lost checkpoints.

Located in the UI at **Advanced Settings** > **Checkpoint Settings** > **Enable checkpointing**, this feature is turned off by default.

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.

Fields for this Source:

- `__final`
- `__inputId`
- `__source`
- `_time`

How to Configure S3 to Send Event Notifications to SQS

For step-by-step instructions, see AWS' Walkthrough: Configure a Bucket for Notifications (SNS Topic and SQS Queue).

1. Create a Standard SQS Queue. Note its ARN.
2. Replace its access policy with one similar to the examples below. To do so, select the queue; and then, in the **Permissions** tab, click: **Edit Policy Document (Advanced)**. (These examples differ only at line 9, showing public access to the SQS queue versus S3-only access to the queue.)
3. In the Amazon S3 console, add a notification configuration to publish events of the `s3:ObjectCreated:*` type to the SQS queue.

Permissive SQS access policy   Restrictive SQS access policy

```
{
 "Version": "2012-10-17",
 "Id": "example-ID",
 "Statement": [
  {
   "Sid": "<SID name>",
   "Effect": "Allow",
   "Principal": {
    "AWS":"*"
   },
   "Action": [
    "SQS:SendMessage"
   ],
   "Resource": "example-SQS-queue-ARN",
   "Condition": {
      "ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:example-bucket-name" }
   }
  }
 ]
}
```

S3 and SQS Permissions

The following permissions are required on the S3 bucket:

- `s3:GetObject`

- `s3:ListBucket`

The following permissions are required on the SQS queue:

- `sqs:ReceiveMessage`
- `sqs:DeleteMessage`
- `sqs:ChangeMessageVisibility`
- `sqs:GetQueueAttributes`
- `sqs:GetQueueUrl`

Temporary Access via SSO Provider

You can use Okta or SAML to provide access to S3 buckets using temporary security credentials.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

How Cribl Stream Pulls Data

Workers poll message from SQS. The call will return messages if they are available, or will time out after 1 second if no messages are available.

Each Worker gets its share of the load from S3. By default, S3 returns a maximum of 1 message in a single poll request. You can change this default in **Max messages**.

Best Practices

Beyond these basics, also see our Amazon S3 Better Practices and Using S3 Storage and Replay guides:

- When Cribl Stream instances are deployed on AWS, use IAM Roles whenever possible.
  - Not only is this safer, but it also makes the configuration simpler to maintain.
- Although optional, we highly recommend that you use a **Filename Filter**.
  - This will ensure that Cribl Stream ingests only files of interest.
  - Ingesting only what's strictly needed improves latency, processing power, and data quality.
- If higher throughput is needed, increase **Advanced Settings** > **Number of Receivers** and/or **Max messages**. However, do note:

    - These are set at `1` by default. As a result, **each** Worker Process, in **each** Cribl Stream Worker Node, will run one receiver consuming one message (that is, S3 file) at a time.
    - Total S3 objects processed at a time per Worker Node = Worker Processes x Number of Receivers x Max Messages
    - Increased throughput implies additional CPU utilization.
- When ingesting large files, tune up the **Visibility Timeout**, or consider using smaller objects.
    - The default value of `600s` works well in most cases, and while you certainly can increase it, we suggest that you also consider using smaller S3 objects.

Troubleshooting Notes

- VPC endpoints for SQS and for S3 might need to be set up in your account. Check with your administrator for details.
- If you're having connectivity issues, but having no problems with the CLI, see if the AWS CLI proxy is in use. Check with your administrator for details.

Troubleshooting Resources

Cribl University offers an Advanced Troubleshooting > Source 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.

15.2.4. Amazon SQS

Cribl Stream supports receiving events from Amazon Simple Queuing Service.

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **No**

Configuring Cribl Stream to Receive Data from Amazon SQS

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Amazon** > **SQS**. 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 SQS Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Queue: The name, URL, or ARN of the SQS queue to read events from. 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

Create queue: If toggled to `Yes`, Cribl Stream will create the queue if it does not exist.

Region: AWS Region where the SQS queue is located. Required, unless the **Queue** entry is a URL or ARN that includes a Region.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down 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 Stream 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 a secret key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS. Follow the **Create** link if you need to configure a key pair.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for SQS: Whether to use Assume Role credentials to access SQS. Defaults to `No`.

AWS account ID: SQS queue owner's AWS account ID. Leave empty if SQS queue is in same AWS account.

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

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

Endpoint: SQS service endpoint. If empty, the endpoint will be automatically constructed from the AWS Region.

Signature version: Signature version to use for signing SQS requests. Defaults to `v4`; `v2` is also available.

Max messages: The maximum number of messages that SQS should return in a poll request. Amazon SQS never returns more messages than this value. (However, fewer messages might be returned.) Acceptable values: `1` to `10`. Defaults to `10`.

Visibility timeout seconds: The duration (in seconds) that the received messages are hidden from subsequent retrieve requests, after they're retrieved by a `ReceiveMessage` request. Defaults to `600`.

Num receivers: The number of receiver processes to run. The higher the number, the better the throughput, at the expense of CPU overhead. Defaults to `3`.

Poll timeout (secs): How long to wait for events before polling again. Minimum `1` second; default `10`; maximum `20`. Short durations increase the number - and thus the cost – of requests sent to AWS. (The UI will show a warning for intervals shorter than `5` seconds.) Long durations increase the time the Source takes to react to configuration changes and system restarts.

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.

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.

Fields for this Source:

- `__final`
- `__inputId`
- `__raw`
- `__receivedTs`
- `__sqsMessageId`
- `__sqsReceiptHandle`
- `_time`

SQS Permissions

The following permissions are needed on the SQS queue:

- `sqs:ReceiveMessage`

- `sqs:DeleteMessage`
- `sqs:GetQueueAttributes`
- `sqs:GetQueueUrl`
- `sqs:CreateQueue` (optional, if and only if you want Cribl Stream to create the queue)

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Troubleshooting Notes

VPC endpoints for SQS might need to be set up in your account. Check with your administrator for details.

How Cribl Stream Pulls Data

Workers poll messages from SQS. The call will return a message if one is available, or will time out after 1 second if no messages are available.

Each Worker gets its share of the load from SQS, and it receives a notification of a file newly added to an S3 bucket. By default, SQS returns a maximum of 10 messages in a single poll request.

15.3. Azure

15.3.1. Azure Blob Storage

Cribl Stream supports receiving data from Azure Blob Storage buckets. Cribl Stream uses Azure Event Grid to receive notifications, via a queue, when new blobs are added to a storage account.

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **YES** Available in: Cribl Stream (LogStream) 2.4.4 and above.

Cribl Stream running on Linux (only) can use this Source to read Parquet files, identified by a `.parquet`, `.parq`, or `.pqt` filename extension.

Restrictions

Cribl Stream supports data ingestion from Azure's **hot** and **cool** access tiers, but not from the **archive** tier – whose stated retrieval lag, up to several hours, cannot guarantee data availability.

This Source supports block blobs, but not append blobs, which can change after they are initially created and the create message is sent. Consider using a Cribl Stream Azure Event Hubs Source if you need to ingest changeable Azure data.

Configuring Cribl Stream to Receive Data from Azure Blob Storage

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Azure** > **Blob Storage**. 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 Azure Blob Storage Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.s

Queue: The queue name from which to read Blob notifications. 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}`.

Authentication Settings

Use the **Authentication method** drop-down 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 Stream 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. 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.

Connection String Format

Either authentication method uses an Azure Storage connection string in this format:

```
DefaultEndpointsProtocol=[http|https];AccountName=<your-account-name>;AccountKey=<y
```

A fictitious example, using Microsoft's recommended HTTPS option, is:

```
DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32;E
```

Optional Settings

Filename filter: Regex matching file names to download and process. Defaults to `.*`, to match all characters.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

This section defines event breaking rulesets that will be applied, in order.

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 Pipelines. 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

Max messages: The maximum number of messages to return in a poll request. Azure queues never return more messages than this value (although they might return fewer messages). Acceptable values: `1` to `32`.

Visibility timeout (secs): The duration (in seconds) that the received messages are hidden from subsequent retrieve requests, after being retrieved by a `ReceiveMessage` request. Defaults to `600` seconds. Maximum allowed value is `604800` seconds (7 days).

> Cribl Stream will automatically extend this timeout until the initial request's files have been processed – notably, in the case of large files that require additional processing time.

Num receivers: The number of receiver processes to run. The higher the number, the better the throughput, at the expense of CPU overhead. Defaults to `1`.

Service period (secs): The interval (in seconds) at which pollers should be validated, and restarted if exited. Defaults to `5` seconds.

Skip file on error: Toggle to `Yes` to skip files that trigger a processing error (e.g., corrupted files). Defaults to `No`, which enables retries after a processing error.

Max Parquet chunk size (MB): Maximum size for each Parquet chunk. Defaults to `5` MB. Valid range is `1` to `100` MB. Cribl Stream stores chunks in the location specified by the `CRIBL_TMP_DIR` environment variable. It removes the chunks immediately after reading them. See Environment Variables.

Parquet chunk download timeout (seconds): The maximum time to wait for a Parquet file's chunk to be downloaded. If a required chunk cannot not be downloaded within this time limit, processing will end. Defaults to `600` seconds. Valid range is `1` second to `3600` seconds (1 hour).

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.

Fields for this Source:

- `__accountName`
- `__final`
- `__inputId`
- `__source`
- `__topic`
- `_time`

The remainder of this topic covers required Azure-side configuration.

Configuring Azure Blob Notifications

This Source needs to receive Azure Event Grid notifications, via a queue, when new blobs are added to a storage account. This queue approach enables Cribl Stream to manage backpressure conditions and retries upon errors.

You will therefore need to enable notifications in the Azure portal. The basic flow is:

File upload → Blob container → Blob Created notification → Azure Queue Storage queue

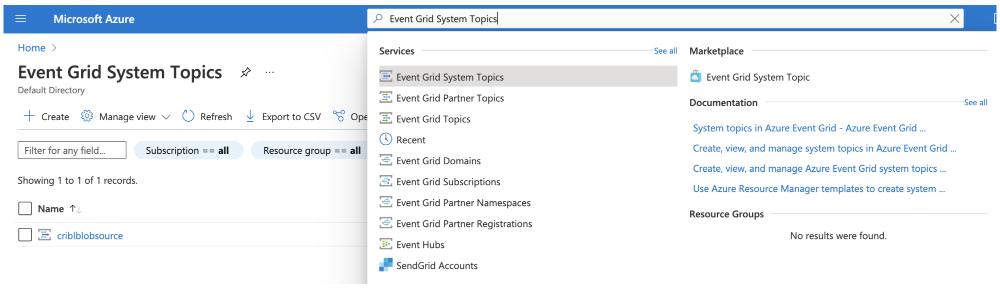
To configure notifications from the Blob storage account in the Azure backend, there are three major steps, outlined below:

1. Create an Event Grid system topic.
2. Create a queue.
3. Configure the generation of storage account notifications when new blobs are uploaded to the queue.

Azure's UI will change over time. Please fall back to Microsoft's Azure Event Grid documentation for up-to-date instructions and screenshots.

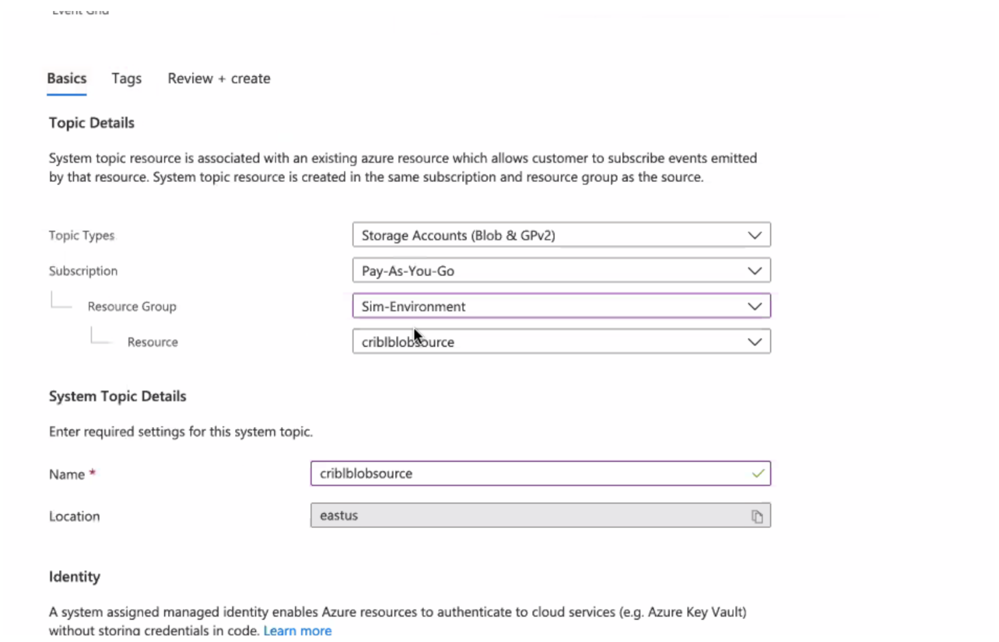
1. Create System Topic

First, you must create a system topic, to which Azure will publish notifications. In the Azure portal, start at **Event Grid System Topics**:



Azure portal > System topics

Select **+Create** to create a new system topic, then set the **Topic Type** to **Storage Account (Blob)**:



Topic Details

System topic resource is associated with an existing azure resource which allows customer to subscribe events emitted by that resource. System topic resource is created in the same subscription and resource group as the source.

Topic Types	Storage Accounts (Blob & GPv2)
Subscription	Pay-As-You-Go
Resource Group	Sim-Environment
Resource	criblblobsource

System Topic Details

Enter required settings for this system topic.

Name *	criblblobsource
Location	eastus

Identity

A system assigned managed identity enables Azure resources to authenticate to cloud services (e.g. Azure Key Vault) without storing credentials in code. Learn more

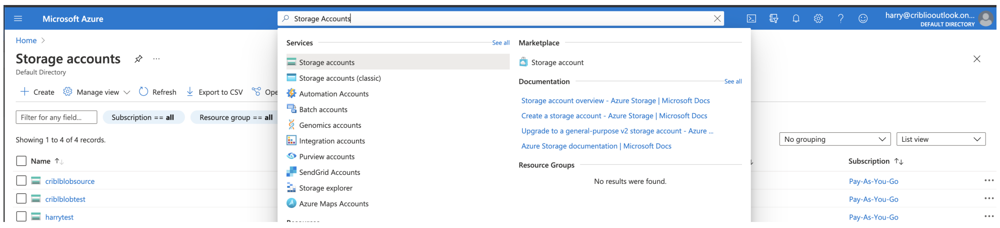
Creating a system topic

In **Subscription** > **Resource Group** > **Resource**, reference the storage account where you want to generate notifications.

Give the topic an arbitrary name that is meaningful to you. (In this example, the name is the same as the storage account.)

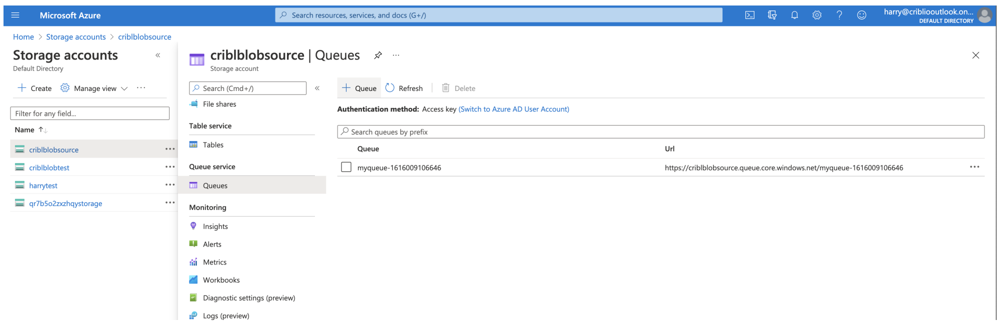
2. Create Storage Queue

Next, navigate to your storage account to create a queue.



Accessing your storage account

Select the storage account for which you would like to set up notifications. Then, in the submenu, select **Queue service** > **Queues**:

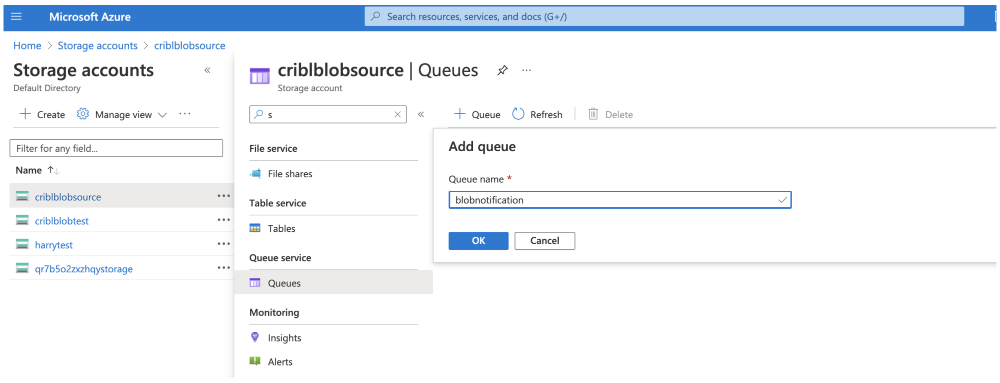


Authentication method: Access key (Switch to Azure AD User Account)

Queue	Url
myqueue-1616009106646	https://criblblobsource.queue.core.windows.net/myqueue-1616009106646

Accessing queues

Select **Create queue**, and give the queue a name that is meaningful to you.

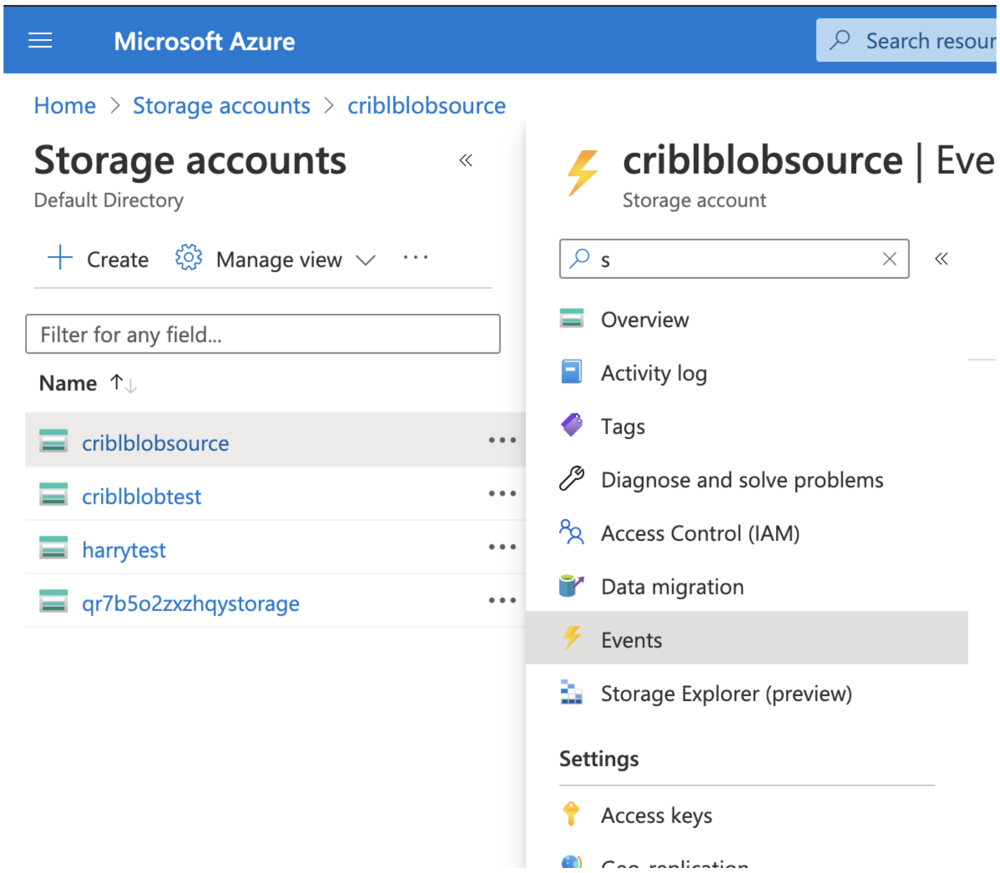


Queue name * blobnotification

Adding a queue

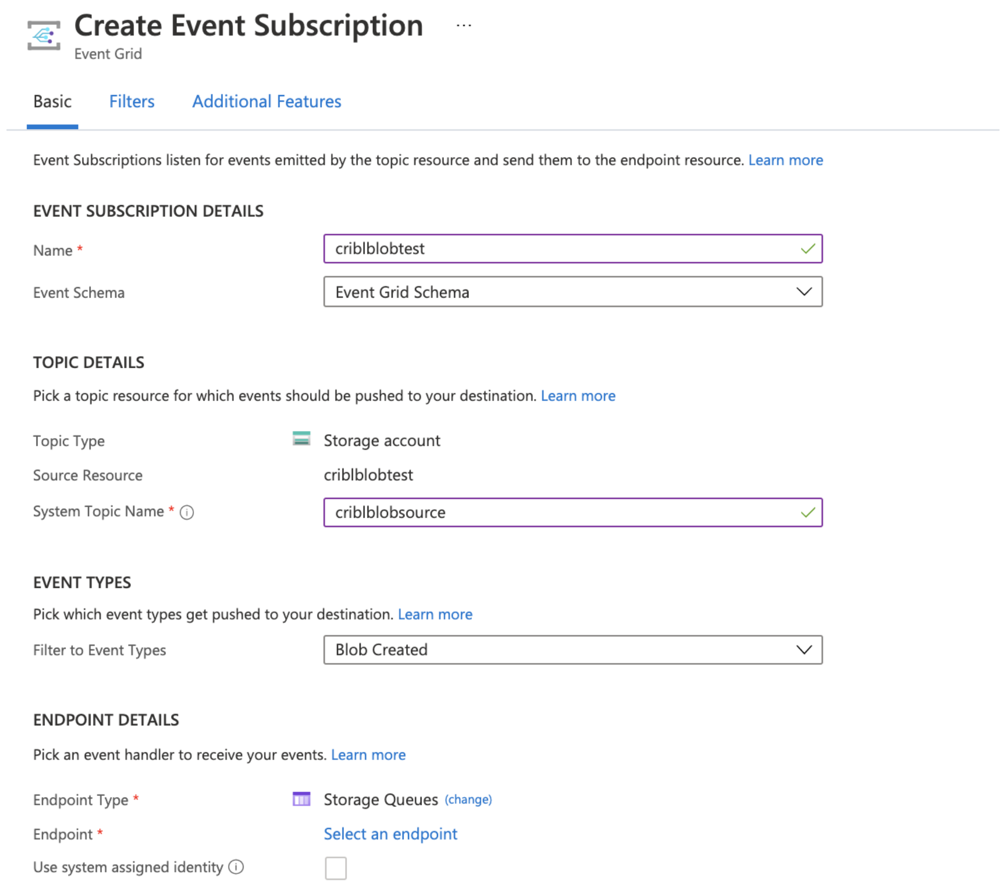
3. Configure Storage Account Notifications

Next, set up the storage account that will publish **Blob Create** notifications to the queue, using the system topic. From the **Storage Accounts** menu, select **Events**:



Accessing your storage account

Then click **+ Event Subscription** to proceed:



Create Event Subscription ...
Event Grid

Basic | Filters | Additional Features

Event Subscriptions listen for events emitted by the topic resource and send them to the endpoint resource. Learn more

EVENT SUBSCRIPTION DETAILS

Name *	criblblobtest
Event Schema	Event Grid Schema

TOPIC DETAILS

Pick a topic resource for which events should be pushed to your destination. Learn more

Topic Type	Storage account
Source Resource	criblblobtest
System Topic Name *	criblblobsource

EVENT TYPES

Pick which event types get pushed to your destination. Learn more

Filter to Event Types	Blob Created

ENDPOINT DETAILS

Pick an event handler to receive your events. Learn more

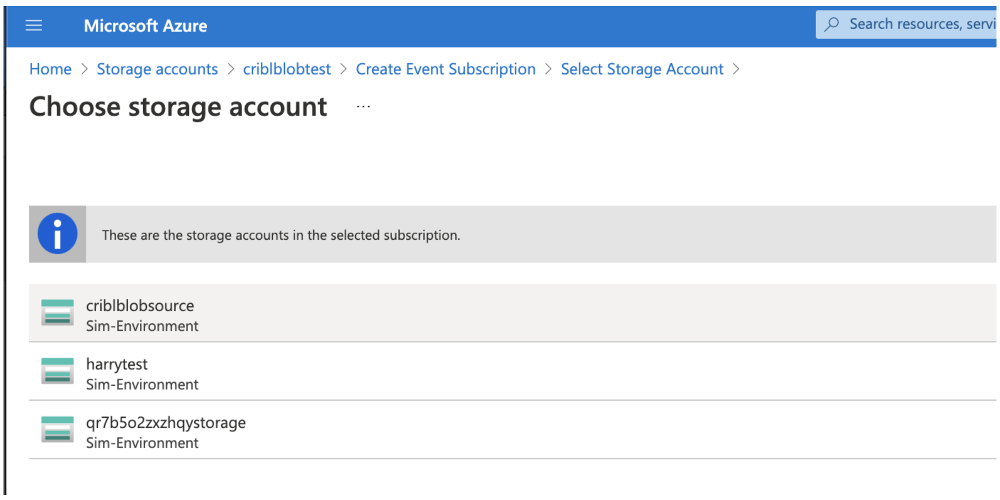
Endpoint Type *	Storage Queues (change)
Endpoint *	Select an endpoint
Use system assigned identity	☐

Creating a subscription.

There are a few things to configure here:

- Enter a **Name** for the subscription.
- In **System Topic Name**, enter the name of the system topic you created in 1. Create System Topic above.
- In **Event Types**, select **Blob Created**, and deselect **Blob Deleted**.
- As the **Endpoint Type**, select **Storage Queues**.
- Click **Select an endpoint**, and click the subscription to use (**Pay-As-You-Go**).

Next, select the storage account on which to add the subscription:


Microsoft Azure

Home > Storage accounts > criblblobtest > Create Event Subscription > Select Storage Account >

Choose storage account

These are the storage accounts in the selected subscription.

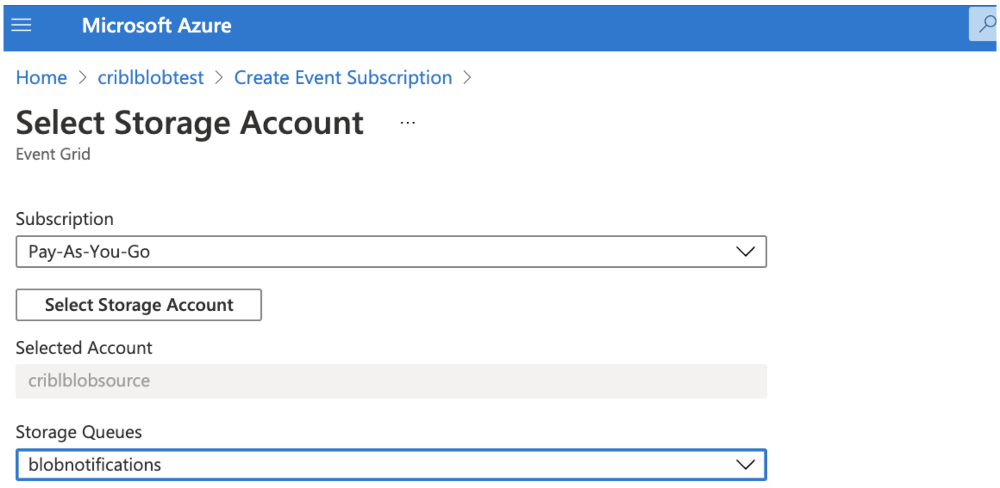
criblblobsource
Sim-Environment

harrytest
Sim-Environment

qr7b5o2zxzhqystorage
Sim-Environment

Choosing the storage account

Select the queue you created in Create Storage Queue above, and click **Confirm Selection** to save the settings.


Microsoft Azure

Home > criblblobtest > Create Event Subscription >

Select Storage Account
Event Grid

Subscription: Pay-As-You-Go

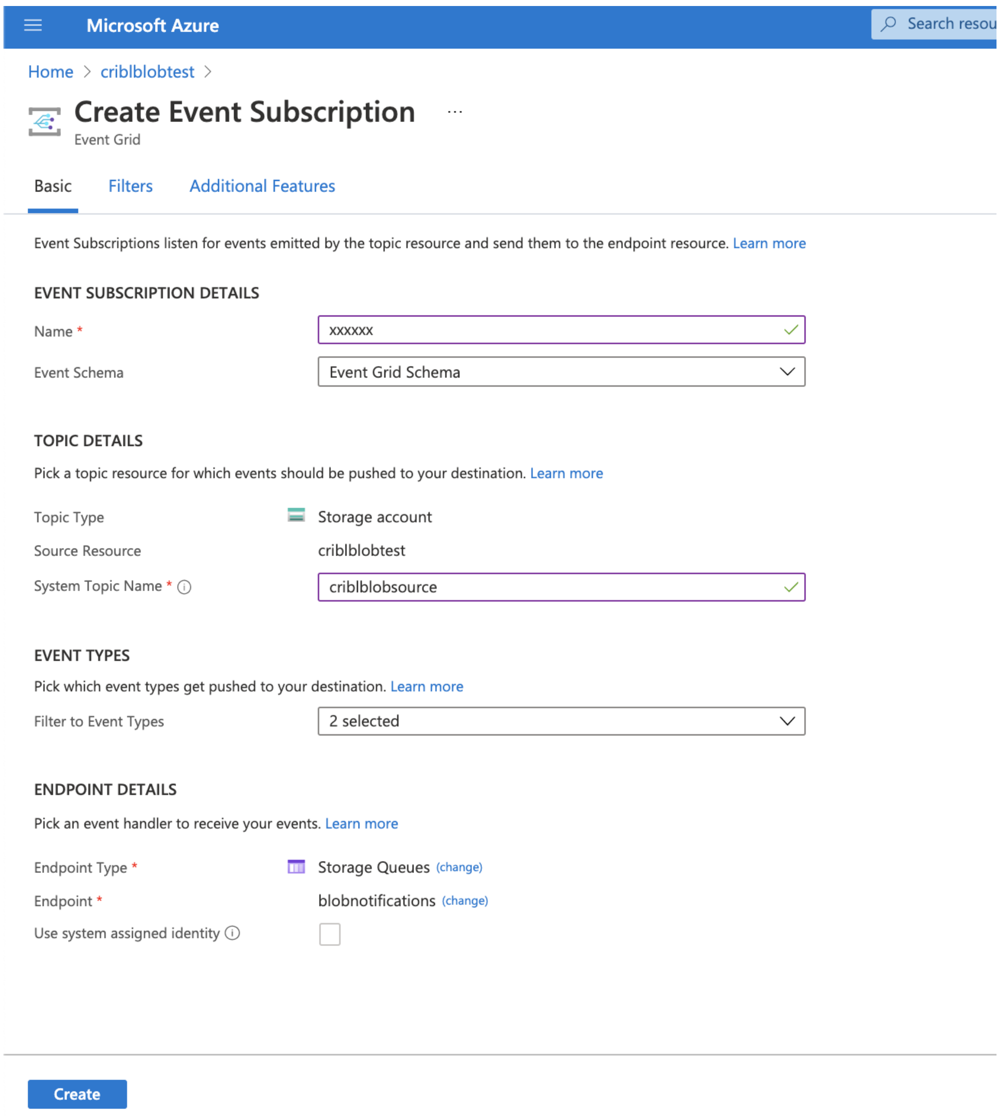
Select Storage Account

Selected Account: criblblobsource

Storage Queues: blobnotifications

Selecting the storage account

To complete the process, click **Create**.


Microsoft Azure

Home > criblblobtest >

Create Event Subscription

Event Grid

Basic | Filters | Additional Features

Event Subscriptions listen for events emitted by the topic resource and send them to the endpoint resource. Learn more

EVENT SUBSCRIPTION DETAILS

Name *	xxxxxx
Event Schema	Event Grid Schema

TOPIC DETAILS

Pick a topic resource for which events should be pushed to your destination. Learn more

Topic Type	Storage account
Source Resource	criblblobtest
System Topic Name *	criblblobsource

EVENT TYPES

Pick which event types get pushed to your destination. Learn more

Filter to Event Types	2 selected

ENDPOINT DETAILS

Pick an event handler to receive your events. Learn more

Endpoint Type *	Storage Queues (change)
Endpoint *	blobnotifications (change)
Use system assigned identity	☐

Creating the subscription

How Cribl Stream Pulls Data

Workers poll messages from Azure Blob Storage using the Azure Event Grid Queue. The call will return a message if one is available, or will time out after 5 seconds if no messages are available.

Each Worker gets its share of the load from Azure Event Grid, and receives a notification of a new file added to an Azure Blob Storage bucket.

By default, the maximum number of messages Azure Event Grid returns in a single poll request is 1 per Worker Process.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

15.3.2. Azure Event Hubs

Cribl Stream supports receiving data records from Azure Event Hubs.

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **No**

Azure Event Hubs uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Stream must receive events directly from senders. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Stream to Receive Data from Azure Event Hubs

From the top nav, click **Manage**, then select a **Worker Group** 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** >] **Azure** > **Event Hubs**. 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** >] **Azure** > **Event Hubs**. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Brokers: List of Event Hubs Kafka brokers to connect to, e.g., `yourdomain.servicebus.windows.net:9093`. Get the hostname from the host portion of the primary or secondary connection string in Shared Access Policies.

Event Hub name: The name of the Event Hub (a.k.a. Kafka Topic) to subscribe to.

Optional Settings

Group ID: The name of the consumer group that includes this Cribl Stream instance. Defaults to `Cribl`.

To prevent excessive Kafka rebalancing and reduced throughput, each **Group ID** that you specify here should be subscribed to only one Kafka Topic – i.e., only to the single Topic you specify in **Event Hub name**. This has a few implications:

- The **Group ID** should be something other than `$Default`, especially if Event Hubs are stored In shared accounts, where the `$Default` group might be subscribed to other Topics.
- The **Group ID** should also be unique for each of your Azure Event Hubs, Kafka, and Confluent Cloud Sources.
- You should configure a **separate** Azure Event Hubs Source for each Group:Topic pair whose events you want to subscribe to.

For details, see Controlling Rebalancing.

From beginning: Whether to start reading from the earliest available data. Relevant only during initial subscription. Defaults to `Yes`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Client Side)

Enabled: Defaults to `Yes`.

Validate server certs: Whether to reject connections to servers without signed certificates. Defaults to `Yes`.

Authentication Settings

Enabled: With the default `Yes` setting, this section's remaining settings are displayed, and all are required settings.

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use with Kafka brokers. Defaults to `PLAIN`, which exposes Basic Authentication options that rely on Azure Event Hubs connection strings. Select `OAUTHBEARER` to enable OAuth Authentication via a different set of options.

Basic Authentication

Selecting the `PLAIN` SASL mechanism provides the options listed in this section.

Username: The username for authentication. For Event Hubs, this should always be `$ConnectionString`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: Use this default option to enter your Event Hubs connection string's primary or secondary key from the Event Hubs workspace. Exposes a **Password** field for this purpose.
- **Secret**: This option exposes a **Password (text secret)** drop-down, in which you can select a stored secret that references an Event Hubs connection string. The secret can reside in Cribl Stream'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=DummyAccessKe
```

OAuth Authentication

Selecting the `OAUTHBEARER` SASL mechanism provides the options listed in this section.

Microsoft Entra ID authentication endpoint: Specifies the Microsoft Entra ID endpoint from which to acquire authentication tokens. Defaults to `https://login.microsoftonline.com`. You can instead select `https://login.microsoftonline.us` or `https://login.partner.microsoftonline.cn`.

Client ID: Enter the `client_id` to pass in the OAuth request parameter.

Tenant identifier: Enter your Microsoft Entra ID subscription's directory ID (tenant ID).

Scope: Enter the scope to pass in the OAuth request parameter. This will be of the form: `https://<Event-Hubs-Namespace-Host-name>/.default`. (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

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Use these settings to fine-tune Cribl Stream's integration with Event Hubs Kafka brokers. For details, see Azure Event Hubs' recommended configuration documentation. If you are unfamiliar with these parameters, contact Cribl Support to understand the implications of changing the defaults.

Heartbeat interval (ms): Expected time between heartbeats to the consumer coordinator when using Kafka's group management facilities. (Corresponds to `heartbeat.interval.ms` in the Kafka domain.) Value must be lower than `sessionTimeout`, and typically should not exceed 1/3 of the `sessionTimeout` value. Defaults to `3000` ms, i.e., 3 seconds.

Session timeout (ms): Timeout used to detect client failures when using Kafka's group management facilities. (Corresponds to `session.timeout.ms` in the Kafka domain.) 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`. Defaults to `30000` ms, i.e., 30 seconds.

Rebalance timeout (ms): Maximum allowed time for each worker to join the group after a rebalance has begun. (Corresponds to `rebalance.timeout.ms` in the Kafka domain.) If this timeout is exceeded, the coordinator broker will remove the worker from the group. Defaults to `60000` ms, i.e., 1 minute.

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.

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Source from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Source to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Offset commit interval (ms): How often, in milliseconds, to commit offsets. If both this field and the **Offset commit threshold** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Offset commit threshold: The number of events that will trigger an offset commit. If both this field and the **Offset commit interval** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Max bytes per partition: The maximum amount of data that the server will return per partition. Must equal or exceed the maximum message size the server allows. (Otherwise, the producer will be unable to send messages larger than the consumer can fetch.) If not specified, defaults to `1048576.`

Max bytes: Maximum amount of bytes to accumulate in the response. The default is `10485760 (10 MB).`

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.

Minimize duplicates: Optionally, toggle to `Yes` to start only one consumer for each topic partition. This reduces duplicates.

> If you observe an excessive number of group rebalances, and/or you observe consumers not regularly pulling messages, try increasing the values of **Heartbeat interval**, **Session timeout**, and **Rebalance timeout**.

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.

Fields for this Source:

- `__inputId`
- `__topicIn` (indicates the Kafka topic that the event came from)
- `__partition`
- `__schemaId` (when using Azure Schema Registry)
- `__key` (when using Schema Registry)
- `__headers` (when using Schema Registry)
- `__keySchemaIdIn` (when using Schema Registry)
- `__valueSchemaIdIn` (when using Schema Registry)

How Cribl Stream Pulls Data

Azure Event Hubs treat all the Worker Nodes as members of a Consumer Group, and each Worker gets its share of the load from Azure Event Hubs. This is the same process as normal Kafka. By default, Workers will poll every 5 seconds. In the case of Leader failure, Worker Nodes will continue to receive data as normal.

Controlling Rebalancing

When you configure multiple Sources that subscribe to different topics, but all belong to the same consumer group, a state change affecting any Source in this consumer group will affect all the other Sources. Examples of state changes include: deploying new configs, adding or removing Worker Processes, or Worker Processes crashing.

Here's an example – three Sources, three different topics, all in one consumer group:

- `Source_1 - Topic_1 - ConsumerGroup1`
- `Source_2 - Topic_2 - ConsumerGroup1`
- `Source_3 - Topic_3 - ConsumerGroup1`

Imagine that Source 1 undergoes a state change event, such as a Worker Process crash. Source 2 and Source 3 will rebalance – stopping data flow until the rebalance completes.

Shared Worker Group Mitigation

If Sources that share a consumer group all deploy as part of the same Worker Group, changes will have smaller side effects than when Sources are spread across different Worker Groups. (Conversely, imagine a configuration where deploying new configs for Worker Group 1 caused rebalancing of topics in worker Worker Group 2. This spillover would be especially undesirable.)

Bottom Line

Changes to any member of a consumer group affect all other members of that consumer group. To prevent this undesired behavior, make sure to use a unique **Group ID** for each Kafka, Confluent Cloud, and Azure Event Hubs Source.

15.4. Google Cloud

15.4.1. Google Cloud Pub/Sub

Cribl Stream supports receiving data records from Google Cloud Pub/Sub, a managed real-time messaging service for sending and receiving messages between applications.

Type: **Pull** | TLS Support: **YES** (secure API) | Event Breaker Support: **No**

Configuring Cribl Stream to Receive Data from Pub/Sub

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Google Cloud** > **Pub/Sub** Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

When working with Google Cloud Pub/Sub, you need to know:

- The project where your credentials originated.
- The project that contains any topic and subscription you want to work with.

If the desired topic and subscription reside in the same project where your credentials originated, describe them using either short names **or** fully-qualified names.

If the desired topic and subscription reside in a different project than the one where your credentials originated, you **must** describe them using fully-qualified names.

Here's an example of a fully-qualified topic name: `projects/my-project-id/topics/my-topic-id`.

> The short name for the same topic would be: `my-topic-id`.

Input ID: Enter a unique name to identify this Pub/Sub Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Topic ID: ID of the Pub/Sub topic from which to receive events.

Subscription ID: ID of the subscription to use when receiving events.

Optional Settings

Create topic: If toggled to `Yes`, Cribl Stream will create the topic on Pub/Sub if it does not exist.

Create subscription: If set to `Yes` (the default), Cribl Stream will create the subscription on Pub/Sub if it does not exist.

Ordered delivery: If toggled to `Yes`, Cribl Stream will receive events in the order that they were added to the queue. (For this to work correctly, the process sending events must have ordering enabled.)

Region: Region to retrieve messages from. 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.)

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down to select a Google authentication method:

Auto: This option uses the environment variables `PUBSUB_PROJECT` and `PUBSUB_CREDENTIALS`, and requires no configuration here.

Manual: With this default option, you use the **Service account credentials** field 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: Use the drop-down to select a key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS.

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 backlog: When the Destination exerts backpressure, this setting limits the number of events that Cribl Stream will queue for processing before it stops retrieving further events. Defaults to `1000` events.

Number of concurrent streams: How many streams to pull messages from at one time. Doubling the value doubles the number of messages this Source pulls from the topic (if available), while consuming more CPU and memory. Defaults to `5`.

Request timeout (ms): Pull request timeout, in milliseconds. Defaults to `60000` ms (i.e., 1 minute).

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.

Fields for this Source:

- `__messageId` – ID of the message from Google.
- `__projectId` – ID of the Google project from which the data was received.
- `__publishTime` – Time at which the event was originally published to the Pub/Sub topic.
- `__subscriptionIn` – The subscription from which the event was received.
- `__topicIn` – The topic from which the event was received.

Google Cloud Roles and Permissions

Your Google Cloud service account should have at least the following roles on subscriptions:

- `roles/pubsub.subscriber`
- `roles/pubsub.viewer` or `roles/viewer`

As an alternative, you can rely on the following discrete permissions, which are included in the above roles:

- `pubsub.snapshots.seek`
- `pubsub.subscriptions.consume`
- `pubsub.topics.attachSubscription`

To enable Cribl Stream's **Create topic** and/or **Create subscription** options, 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.

How Cribl Stream Pulls Data

Pub/Sub treats all the Worker Nodes as members of a Consumer Group, and each Worker gets its share of the load from Pub/Sub. This is the same process as normal Kafka. By default, Workers will poll every 1 minute. In the case of Leader failure, Worker Nodes will continue to receive data as normal.

15.5. Kafka

15.5.1. Kafka

Cribl Stream supports receiving data records from a Kafka cluster. As of Cribl Stream v.3.3, this Source automatically detects compressed data in `Gzip`, `Snappy`, or `LZ4` format.

Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **No**

Kafka uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Stream must receive events directly from senders. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Stream to Receive Data from Kafka Topics

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Kafka**. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Brokers: List of Kafka brokers to use, e.g., `localhost:9092`.

Topics: Enter the name(s) of topics to subscribe to. Press `Enter/Return` between multiple entries.

To optimize performance and prevent excessive rebalancing, Cribl recommends subscribing each Kafka Source to only one topic. To subscribe to multiple topics, consider creating a dedicated Kafka Source for each one.

> For the same reason, the **Group ID** (below) should be unique for each of your Kafka, Confluent Cloud, and Azure Event Hubs Sources. For details, see Controlling Rebalancing.

Optional Settings

Group ID: The name of the consumer group to which this Cribl Stream instance belongs.

From beginning: Whether to start reading from the earliest available data. Relevant only during initial subscription. Defaults to `Yes`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Client Side)

Enabled: defaults to `No`. When toggled to `Yes`:

Autofill?: This setting is experimental.

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`.

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

SASL mechanism: Use this drop-down to select the SASL authentication mechanism to use. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these 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 keytab 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`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. See Kafka Authentication with Kerberos for further detail.

Schema Registry

This section governs Kafka Schema Registry authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: Defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (E.g., `http://<hostname>:8081`.)

TLS enabled: When toggled to `Yes`, displays the following TLS settings for the Schema Registry.

These have 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

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Use these settings to fine-tune Cribl Stream's integration with Kafka topics. If you are unfamiliar with these parameters, contact Cribl Support to understand the implications of changing the defaults.

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. Defaults to `3000` ms, i.e., 3 seconds. For details, see the Kafka documentation.

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`. Defaults to `30000` ms, i.e., 30 seconds. For details, see the Kafka documentation.

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. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. For details, see the Kafka documentation.

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Source from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Source to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Offset commit interval (ms): How often, in milliseconds, to commit offsets. If both this field and the **Offset commit threshold** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Offset commit threshold: The number of events that will trigger an offset commit. If both this field and the **Offset commit interval** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Max bytes per partition: The maximum amount of data that the server will return per partition. Must equal or exceed the maximum message size the server allows. (Otherwise, the producer will be unable to send messages larger than the consumer can fetch.) If not specified, defaults to `1048576`.

Max bytes: Maximum amount of bytes to accumulate in the response. The default is `10485760 (10 MB)`.

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.

> If you observe an excessive number of group rebalances, and/or you observe consumers not regularly pulling messages, try increasing the values of **Heartbeat interval**, **Session timeout**, and **Rebalance timeout**.

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.

Connecting to Kafka

> ### Leave the TLS Settings > Server name (SNI) field blank
>
> In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.
>
> Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.
2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service – among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Internal Fields

Cribl 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.

Fields for this Source:

- `__final`
- `__headers` (when present in the record)
- `__inputId`
- `__key` (when using Schema Registry)
- `__keySchemaIdIn` (when using Schema Registry)
- `__origTime`
- `__partition`
- `__raw`
- `__schemaId` (when using Schema Registry)

- `__topicIn` (indicates the Kafka topic that the event came from; see `__topicOut` in our Kafka Destination documentation)
- `__valueSchemaIdIn` (when using Schema Registry)
- `_raw`
- `_time`

How Cribl Stream Pulls Data

Kafka treats all the Worker Nodes as members of a Consumer Group, and Kafka manages each Node's data load. By default, Workers will poll every 5 seconds. In the case of Leader failure, Worker Nodes will continue to receive data as normal.

How Cribl Stream Handles the `_time` Field

Events that the Kafka Source emits always contain a `_time` field, and sometimes also an `__origTime` field. Here's how Cribl Stream determines what to send:

- If the incoming Kafka message contains no `_time` field, the message's timestamp becomes the value of the emitted event's `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is a timestamp in UNIX epoch time, that timestamp becomes the value of the emitted event's `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is **not** a timestamp in UNIX epoch time (e.g., an ISO or UTC timestamp), that becomes the value of the emitted event's `__origTime` field, **and** the message's timestamp becomes the value of the emitted event's `_time` field.

Controlling Rebalancing

When you configure multiple Sources that subscribe to different topics, but all belong to the same consumer group, a state change affecting any Source in this consumer group will affect all the other Sources. Examples of state changes include: deploying new configs, adding or removing Worker Processes, or Worker Processes crashing.

Here's an example – three Sources, three different topics, all in one consumer group:

- `Source_1 - Topic_1 - ConsumerGroup1`
- `Source_2 - Topic_2 - ConsumerGroup1`
- `Source_3 - Topic_3 - ConsumerGroup1`

Imagine that Source 1 undergoes a state change event, such as a Worker Process crash. Source 2 and Source 3 will rebalance – stopping data flow until the rebalance completes.

Shared Worker Group Mitigation

If Sources that share a consumer group all deploy as part of the same Worker Group, changes will have smaller side effects than when Sources are spread across different Worker Groups. (Conversely, imagine a configuration where deploying new configs for Worker Group 1 caused rebalancing of topics in worker Worker Group 2. This spillover would be especially undesirable.)

Bottom Line

Changes to any member of a consumer group affect all other members of that consumer group. To prevent this undesired behavior, make sure to use a unique **Group ID** for each Kafka, Confluent Cloud, and Azure Event Hubs Source.

15.5.2. Confluent Cloud

Cribl Stream supports receiving Kafka topics from the Confluent Cloud managed Kafka platform. As of Cribl Stream v.3.3, this Source automatically detects compressed data in `Gzip`, `Snappy`, or `LZ4` format.

Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **No**

Confluent Cloud uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Stream must receive events directly from senders. You might need to adjust your firewall rules to allow this traffic.

Your Confluent Cloud permissions should include `Consumer Group: Read`.

Ingesting Kafka Topics from Confluent Cloud

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Confluent Cloud**. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Brokers: List of Confluent Cloud brokers to use, e.g., `myAccount.confluent.cloud:9092`.

Topics: Enter the name(s) of topics to subscribe to. Press `Enter/Return` between multiple entries.

To optimize performance and prevent excessive rebalancing, Cribl recommends subscribing each Confluent Cloud Source to only one topic. To subscribe to multiple topics, consider creating a dedicated Confluent Cloud Source for each one.

> For the same reason, the **Group ID** (below) should be unique for each of your Confluent Cloud, Kafka, and Azure Event Hubs Sources. For details, see Controlling Rebalancing.

Optional Settings

Group ID: The name of the consumer group to which this Cribl Stream instance belongs.

From beginning: Whether to start reading from the earliest available data. Relevant only during initial subscription. Defaults to `Yes`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Client Side)

Enabled: defaults to `No`. When toggled to `Yes`:

Autofill?: This setting is experimental.

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`.

Server name (SNI): Leave this field blank. See Connecting to Kafka below.

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.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

- **SASL mechanism**: Select the SASL (Simple Authentication and Security Layer) authentication mechanism to use. Defaults to `PLAIN`. `SCRAM-SHA-256`, `SCRAM-SHA-512`, and `GSSAPI/Kerberos` are also available. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these SASL mechanisms, select one of the following buttons:

Manual: Displays **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 keytab 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`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. See Kafka Authentication with Kerberos for further detail.

Schema Registry

This section governs Confluent 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`.)

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)**: Leave this field blank. See About Kafka and TLS below.
- **Minimum TLS version**: Optionally, select the minimum TLS version to use when connecting.
- **Maximum TLS version**: Optionally, select the maximum TLS version to use when connecting.
- **Certificate name**: The name of the predefined certificate.
- **CA certificate path**: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.
- **Private key path (mutual auth)**: Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Certificate path (mutual auth)**: Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Passphrase**: Passphrase to use to decrypt private key.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Use these settings to fine-tune Cribl Stream's integration with Kafka topics. If you are unfamiliar with these parameters, contact Cribl Support to understand the implications of changing the defaults.

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. Defaults to `3000` ms, i.e., 3 seconds. For details, see the Kafka documentation.

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`. Defaults to `30000` ms, i.e., 30 seconds. For details, see the Kafka documentation.

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. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. For details, see the Kafka documentation.

Request timeout (ms): Maximum time to wait for a successful request. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Source from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Source to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Offset commit interval (ms): How often, in milliseconds, to commit offsets. If both this field and the **Offset commit threshold** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Offset commit threshold: The number of events that will trigger an offset commit. If both this field and the **Offset commit interval** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Max bytes per partition: The maximum amount of data that the server will return per partition. Must equal or exceed the maximum message size the server allows. (Otherwise, the producer will be unable to send messages larger than the consumer can fetch.) If not specified, defaults to `1048576`.

Max bytes: Maximum amount of bytes to accumulate in the response. The default is `10485760 (10 MB)`.

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.

> If you observe an excessive number of group rebalances, and/or you observe consumers not regularly pulling messages, try increasing the values of **Heartbeat interval**, **Session timeout**, and **Rebalance timeout**.

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.

Connecting to Kafka

> ### Leave the TLS Settings > Server name (SNI) field blank
>
> In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.
>
> Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation. This is especially important for Confluent Cloud Dedicated clusters, which rely on SNI – as managed by the Kafka library – for routing.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds.

Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.
2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service – among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Internal Fields

Cribl 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.

Fields for this Source:

- `__inputId`
- `__topicIn` (indicates the Confluent Cloud topic that the event came from; see `__topicOut` in our [Confluent Cloud Destination](destinations-Confluent Cloud) documentation)
- `__partition`
- `__schemaId` (when using Schema Registry)

How Cribl Stream Pulls Data

Confluent Cloud treats all the Worker Nodes as members of a Consumer Group, and Confluent Cloud manages each Node's data load. By default, Workers will poll every 5 seconds. In the case of Leader failure, Worker Nodes will continue to receive data as normal.

How Cribl Stream Handles the `_time` Field

Events that the Kafka Source emits always contain a `_time` field, and sometimes also an `__origTime` field. Here's how Cribl Stream determines what to send:

- If the incoming Kafka message contains no `_time` field, the message's timestamp becomes the value of the emitted event's `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is a timestamp in UNIX epoch time, that timestamp becomes the value of the emitted event's `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is **not** a timestamp in UNIX epoch time (e.g., an ISO or UTC timestamp), that becomes the value of the emitted event's `__origTime` field, **and** the message's timestamp becomes the value of the emitted event's `_time` field.

Controlling Rebalancing

When you configure multiple Sources that subscribe to different topics, but all belong to the same consumer group, a state change affecting any Source in this consumer group will affect all the other Sources. Examples of state changes include: deploying new configs, adding or removing Worker Processes, or Worker Processes crashing.

Here's an example – three Sources, three different topics, all in one consumer group:

- `Source_1 - Topic_1 - ConsumerGroup1`
- `Source_2 - Topic_2 - ConsumerGroup1`
- `Source_3 - Topic_3 - ConsumerGroup1`

Imagine that Source 1 undergoes a state change event, such as a Worker Process crash. Source 2 and Source 3 will rebalance – stopping data flow until the rebalance completes.

Shared Worker Group Mitigation

If Sources that share a consumer group all deploy as part of the same Worker Group, changes will have smaller side effects than when Sources are spread across different Worker Groups. (Conversely, imagine a configuration where deploying new configs for Worker Group 1 caused rebalancing of topics in worker Worker Group 2. This spillover would be especially undesirable.)

Bottom Line

Changes to any member of a consumer group affect all other members of that consumer group. To prevent this undesired behavior, make sure to use a unique **Group ID** for each Kafka, Confluent Cloud, and Azure Event Hubs Source.

15.5.3. Amazon MSK

Cribl Stream supports receiving data records from an Amazon Managed Streaming for Apache Kafka (MSK) cluster. This Source automatically detects compressed data in `Gzip`, `Snappy`, or `LZ4` format.

Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **No**

Kafka uses a binary protocol over TCP. It does not support HTTP proxies, so Cribl Stream must receive events directly from senders. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Stream to Receive Data from Kafka Topics

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

In the **QuickConnect** UI: Click **Add Source**. From the resulting drawer's tiles, select [**Pull** >] **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** > **Sources**. From the resulting page's tiles or left nav, select [**Pull** >] **Amazon MSK**. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Brokers: List of Kafka brokers to connect to, e.g., `kafkaBrokerHost:9092`. Specify hostname and port, e.g., `mykafkabroker:9092`, or just hostname, in which case Stream will assign port 9092.

Topics: Enter the name(s) of topics to subscribe to. Press `Enter/Return` between multiple entries.

To optimize performance and prevent excessive rebalancing, Cribl recommends subscribing each Kafka Source to only one topic. To subscribe to multiple topics, consider creating a dedicated Kafka Source for each one.

For the same reason, the **Group ID** (below) should be unique for each of your Kafka, Confluent Cloud, and Azure Event Hubs Sources. For details, see Controlling Rebalancing.

Region: Select the name of the AWS Region where your Amazon MSK cluster is located.

Optional Settings

Group ID: The name of the consumer group to which this Cribl Stream instance belongs.

From beginning: Toggle this off if you want Cribl Stream, upon subscribing to a topic for the first time, to read new messages only. Otherwise Cribl Stream will read all messages available on the server, starting from the oldest.

Tags: Optionally, add tags that you can use for filtering and grouping in the Cribl Stream UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

TLS Settings (Client Side)

For Amazon MSK Sources and Destinations:

- IAM is the only type of authentication that Cribl Stream 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): Leave this field blank. See Connecting to Kafka below.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Authentication

Use the **Authentication method** drop-down 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 Stream 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for MSK: Toggle on to use Assume Role credentials to access MSK.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Use these settings to fine-tune Cribl Stream's integration with Kafka topics. If you are unfamiliar with these parameters, contact Cribl Support to understand the implications of changing the defaults.

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. Defaults to `3000` ms, i.e., 3 seconds. For details, see the Kafka documentation.

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`. Defaults to `30000` ms, i.e., 30 seconds. For details, see the Kafka documentation.

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. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. For details, see the Kafka documentation.

Request timeout (ms): Maximum time to wait for Kafka to respond to a request. Defaults to `60000` ms, i.e., 1 minute. For details, see the Kafka documentation.

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. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Source from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Source to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Offset commit interval (ms): How often, in milliseconds, to commit offsets. If both this field and the **Offset commit threshold** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Offset commit threshold: How many events are needed to trigger an offset commit. If both this field and the **Offset commit interval** are empty, Cribl Stream will commit offsets after each batch. If both fields are set, Cribl Stream will commit offsets when either condition is met.

Max bytes per partition: Maximum amount of data that Kafka will return per partition, per fetch request. Must equal or exceed the maximum message size (maxBytesPerPartition) that Kafka is configured to allow. Otherwise, Cribl Stream can get stuck trying to retrieve messages. Defaults to `1048576 (1 MB)`.

Max bytes: Maximum number of bytes that Kafka will return per fetch request. Defaults to `10485760 (10 MB)`.

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.

> If you observe an excessive number of group rebalances, and/or you observe consumers not regularly pulling messages, try increasing the values of **Heartbeat interval**, **Session timeout**, and **Rebalance timeout**.

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.

Connecting to Kafka

Leave the TLS Settings > Server name (SNI) field blank

In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.
2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service –

among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Internal Fields

Cribl 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.

Fields for this Source:

- `__inputId`
- `__topicIn` (indicates the Kafka topic that the event came from; see `__topicOut` in our Kafka Destination documentation)
- `__partition`
- `__schemaId` (when using Schema Registry)
- `__key` (when using Schema Registry)
- `__headers` (when using Schema Registry)
- `__keySchemaIdIn` (when using Schema Registry)
- `__valueSchemaIdIn` (when using Schema Registry)

How Cribl Stream Pulls Data

Kafka treats all the Worker Nodes as members of a Consumer Group, and Kafka manages each Node’s data load. By default, Workers will poll every 5 seconds. In the case of Leader failure, Worker Nodes will continue to receive data as normal.

How Cribl Stream Handles the `_time` Field

Events that the Kafka Source emits always contain a `_time` field, and sometimes also an `__origTime` field. Here’s how Cribl Stream determines what to send:

- If the incoming Kafka message contains no `_time` field, the message’s timestamp becomes the value of the emitted event’s `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is a timestamp in UNIX epoch time, that timestamp becomes the value of the emitted event’s `_time` field.
- If the incoming Kafka message contains a `_time` field whose value is **not** a timestamp in UNIX epoch time (e.g., an ISO or UTC timestamp), that becomes the value of the emitted event’s `__origTime` field,

and the message's timestamp becomes the value of the emitted event's `_time` field.

Controlling Rebalancing

When you configure multiple Sources that subscribe to different topics but all belong to the same consumer group, a state change affecting any Source in this consumer group will affect all the other Sources. Examples of state changes include: deploying new configs, adding or removing Worker Processes, and Worker Processes crashing.

Here's an example – three Sources, three different topics, all in one consumer group:

- `Source_1 - Topic_1 - ConsumerGroup1`
- `Source_2 - Topic_2 - ConsumerGroup1`
- `Source_3 - Topic_3 - ConsumerGroup1`

Imagine that Source 1 undergoes a state change event, such as a Worker Process crash. Source 2 and Source 3 will rebalance – stopping data flow until the rebalance completes.

Shared Worker Group Mitigation

If Sources that share a consumer group all deploy as part of the same Worker Group, changes will have smaller side effects than when Sources are spread across different Worker Groups. (Conversely, imagine a configuration where deploying new configs for Worker Group 1 caused rebalancing of topics in Worker Worker Group 2. This spillover would be especially undesirable.)

Bottom Line

Changes to any member of a consumer group affect all other members of that consumer group. To prevent this undesired behavior, make sure to use a unique **Group ID** for each Kafka, Confluent Cloud, Amazon MSK, and Azure Event Hubs Source.

15.6. Office 365

15.6.1. Office 365 Activity

Cribl Stream supports receiving data from the Office 365 Management Activity API. This facilitates analyzing actions and events on Microsoft Entra ID, Exchange, and SharePoint, along with global auditing and Data Loss Prevention data.

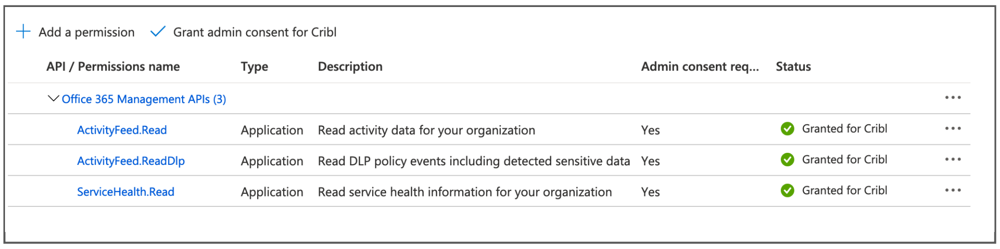
Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **YES**

TLS is enabled via the HTTPS protocol on this Source's underlying REST API.

Microsoft Entra ID Permissions

In Microsoft Entra ID, the application representing your Cribl Stream instance must be granted the following permissions to pull data. Each permission's **Type** must be `Application` – `Delegated` is not sufficient:

- `ActivityFeed.Read` – Required for all Content Types except `DLP.All`.
- `ActivityFeed.ReadDlp` – Required for the `DLP.All` Content Type.



+ Add a permission ✓ Grant admin consent for Cribl

API / Permissions name	Type	Description	Admin consent req...	Status
Office 365 Management APIs (3)				
ActivityFeed.Read	Application	Read activity data for your organization	Yes	✓ Granted for Cribl
ActivityFeed.ReadDlp	Application	Read DLP policy events including detected sensitive data	Yes	✓ Granted for Cribl
ServiceHealth.Read	Application	Read service health information for your organization	Yes	✓ Granted for Cribl

Registered application permissions

Office 365 Subscriptions

Cribl Stream does not support starting/stopping Office 365 subscriptions. You can start subscriptions either via another Office 365 API client, or simply via `curl` commands. We document the `curl` command method below in Starting Content Subscriptions.

Configuring Cribl Stream to Receive Data from the Activity API

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Office 365**. 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** >] *Office 365**. 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 Office 365 Activity definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Tenant ID: Enter the Office 365 Azure tenant ID.

App ID: Enter the Office 365 Azure application ID.

Subscription Plan: Select the Office 365 subscription plan for your organization. Options include:

- **Office 365 Enterprise**
- **Office 365 GCC**
- **Office 365 GCC High**
- **Office 365 DoD**

Authentication Settings

Authentication method: Select one of the following buttons.

- **Manual**: This default option provides a **Client secret** field, where you directly enter the required Office 365 Azure client secret.
- **Secret**: This option instead exposes a **Client secret (text secret)** drop-down, from which you select a stored text secret to authenticate with. Click **Create** to configure a new secret.

Optional Settings

Publisher identifier: Use in API requests as described here. If not defined, defaults to Microsoft Office 365 tenant ID.

Content Types: See the Content Types section below.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Content Types

Here, you can configure polling independently for the following types of audit data from the Office 365 Management Activity API:

- **Active Directory**
- **Exchange**
- **SharePoint**
- **General**: All workloads not included in the above content types
- **DLP.All**: Data Loss Prevention events only, for all workloads

For each of these content types, the **Content Types** table provides the following controls:

Interval Description: This column is informational only.

Interval: Optionally, override the default polling interval. See About Polling Intervals below.

Log Level: Set the verbosity level to one of `debug`, `info` (the default), `warn`, or `error`.

Enabled: Toggle this to `Yes` for each service that you want to poll.

About Polling Intervals

To poll the Office 365 Management Activity API, Cribl Stream uses the **Interval** field's value to establish the search date range and the cron schedule (e.g.: `*/${interval} * * * *`).

Therefore, intervals set in minutes must divide evenly into 60 minutes to create a predictable schedule. Dividing 60 by intervals like `1`, `2`, `3`, `4`, `5`, `6`, `10`, `12`, `15`, `20`, or `60` itself yields an integer, so you can enter any of these values.

Cribl Stream will reject intervals like `23`, `42`, or `45`, or `75` – which would yield non-integer results, meaning unpredictable schedules.

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.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Advanced Settings

Ingestion lag (minutes): How far back into the past the Office 365 Activity API should look for events to retrieve. This is necessary because there can be a lag of 60 to 90 minutes (or longer) before Office 365 events become available via the API. The default value of `0` means don't look back. When it's very important to avoid missing events, consider running multiple overlapping jobs, as described below.

Keep alive time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `30`.

Worker timeout (periods): The number of **Keep alive time** periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

Request timeout (secs): The maximum time period for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `300` (i.e., 5 minutes). Enter `0` to disable timeout metering.

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. Defaults to `0`.

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 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.

Fields for this Source:

- `__collectible`
- `__final`
- `__inputId`
- `__isBroken`
- `__source`

Starting Content Subscriptions

Content subscriptions (a different concept from the O365 subscription plans) are required in order for Cribl Stream to be able to begin retrieving O365 data. There is a separate subscription required for each Content Type. If you are using an existing Azure-registered application ID that already has subscriptions started, then you can ignore this section. But if you are:

- Using a newly registered application ID, and therefore never had any subscriptions started, or
- Reusing an application ID that had subscriptions started, but are currently stopped

…then you will need to use this procedure to manually start the necessary subscriptions. Follow either of the two methods below, using (respectively) PowerShell or `curl`.

Using PowerShell

This sample PowerShell script will enable all subscriptions for you. Update the appropriate variables as required:

```
# Create app of type Web app / API in Microsoft Entra ID, generate a Client Secret
# Get the tenant GUID from Properties | Directory ID under the Microsoft Entra ID s
$AppID = "<APP_ID>"
$ClientSecret = "<CLIENT_SECRET>"
$TenantID = "<TENANT_ID>"
$loginURL = "https://login.microsoftonline.com/"

# For $resource, use one of these endpoint values based on your subscription plan:
# * Enterprise - manage.office.com
# * GCC - manage-gcc.office.com
# * GCC High - manage.office365.us
# * DoD - manage.protection.apps.mil
$resource = "https://manage.office.com"

$body = @{grant_type="client_credentials";resource=$resource;client_id=$AppID;clie
$oauth = Invoke-RestMethod -Method Post -Uri $loginURL/$TenantID/oauth2/token?api-
$headerParams = @{'Authorization'="$($oauth.token_type) $($oauth.access_token)"}

Invoke-WebRequest -Headers $headerParams -Uri "$resource/api/v1.0/$TenantID/activi

Invoke-WebRequest -Method Post -Headers $headerParams -Uri "$resource/api/v1.0/$Te
Invoke-WebRequest -Method Post -Headers $headerParams -Uri "$resource/api/v1.0/$Te
Invoke-WebRequest -Method Post -Headers $headerParams -Uri "$resource/api/v1.0/$Te
Invoke-WebRequest -Method Post -Headers $headerParams -Uri "$resource/api/v1.0/$Te
Invoke-WebRequest -Method Post -Headers $headerParams -Uri "$resource/api/v1.0/$Te
```

Using curl

This is a two-step process. The first command obtains an auth token, which is used in the second command to actually start the subscription. To execute these commands, you'll need the same information (i.e., client secret, application ID, and tenant ID) that you already require to configure this Source in Cribl Stream's GUI. Replace those three variables as appropriate in the commands below.

1. `curl -d "client_secret=<client secret>&resource=https://manage.office.com&client_id=<app id>&grant_type=client_credentials" -X POST https://login.windows.net/<tenant id>/oauth2/token`
2. `curl -d "" -H "Authorization: Bearer <access token>" -X POST https://manage.office.com/api/v1.0/<tenant id>/activity/feed/subscriptions/start?contentType=<content_type_name>`

Here is an example of each command executed and expected output:

Example Command #1

```
$ curl -d
"client_secret=abcdefghijklmnopqrstuvwxyz12345678&resource=https://manage.office.com&c
ffff-ffff-ffff-aaaaaaaaaaaa&grant_type=client_credentials" -X POST
https://login.windows.net/12345678-aaaa-4233-cccc-160c6c30154a/oauth2/token
```

Output:

```
{"token_type":"Bearer","expires_in":"3599","ext_expires_in":"3599","expires_on":"16220
JWT here...MRDvw"}
```

Example Command #2

```
$ curl -d "" -H "Authorization: Bearer eyJ0...long JWT here...MRDvw" -X POST
"https://manage.office.com/api/v1.0/12345678-aaaa-4233-cccc-
160c6c30154a/activity/feed/subscriptions/start?
contentType=Audit.AzureActiveDirectory"
```

Output:

```
{"contentType":"Audit.AzureActiveDirectory","status":"enabled","webhook":null}
```

Note there is no output when executing this second command with a `stop` operation.

You'll need to execute the second command for each Content Type whose logs you wish to collect. Use the exact strings below to specify Content Types in that command:

- `Audit.AzureActiveDirectory`
- `Audit.Exchange`
- `Audit.SharePoint`

- `Audit.General`
- `DLP.All`

How Cribl Stream Pulls Data

The Office 365 Activity Source retrieves data using Cribl Stream scheduled Collection jobs, which include Discover and Collection phases. The Discover phase task returns the URL of the content to collect.

In the Source's **General Settings** > **Content Types** > **Interval** column, you configure the polling schedule for each Content Type independently. Each Content Type that you enable gets its own separate scheduled job.

The job scheduler spreads the Collection tasks across all available Workers. The collected content is paginated, so the collection phase might include multiple calls to fetch data.

Viewing Scheduled Jobs

Once you've configured and saved the Source, you can view the Collection jobs' results by reopening the Source's config modal and clicking its **Job Inspector** tab.

You can also view these jobs (among scheduled jobs for other Collectors and Sources) in the **Monitoring** > **System** > **Job Inspector** > **Currently Scheduled** tab.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a "stuck", "orphaned," or "zombie" job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream's logs.

To keep stuck jobs from running excessively long:

- First, try setting **Advanced** > **Timeout (secs)** to a duration shorter than the default of `300` seconds (5 minutes).
- If adjusting **Timeout (secs)** does not fix the problem, try the global setting **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration. You'll find **Job Timeout** among the task manifest and buffering limits.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Multiple Overlapping Jobs

You can schedule multiple Collection jobs where each job uses a different ingestion lag, e.g., three different jobs with **Ingestion lag** set to `120` (two hours), `720` (twelve hours), and `5760` (four days), respectively.

Although this approach will collect data over partially overlapping time ranges – meaning that you'll have duplicate events – it can improve the chances that you'll catch "straggler" events affected by the known lag times in the Office 365 Management Activity API.

15.6.2. Office 365 Message Trace

Cribl Stream supports receiving Office 365 Message Trace data. This mail-flow metadata can be used to detect and report on malicious activity including bulk emails, spoofed-domain emails, and data exfiltration.

Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **YES**

TLS is enabled via the HTTPS protocol on this Source's underlying REST API.

Office 365 Setup

At a minimum, your Office 365 service account should include a role with `Message Tracking` and `View-Only Recipients` permissions, assigned to the Office 365 user that will integrate with Cribl Stream. Assign these permissions in the Exchange admin center (https://admin.exchange.microsoft.com).

Modern Authentication (OAuth 2.0) Setup

If you plan to use OAuth, OAuth Secret, or OAuth Certificate authentication, your Office 365 setup must include at least one role with the corresponding required permissions:

1. In the Azure portal, create a Microsoft Entra ID App Registration. (For details, see documentation from Microsoft, Splunk, or Splunkbase.)
2. Assign the application at least one Azure AD role that will enable it to access the Reporting Web Service:
    - Global Reader
    - Global Administrator
    - Exchange Administrator

Make sure that at least one role includes the `ReportingWebService.Read.All` permission. For detailed steps, see Microsoft's Assign Azure AD Roles to Users topic.

Configuring Cribl Stream to Receive Office 365 Message Trace Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Office 365** > **Message Trace**.

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** >] **Office 365** > **Message Trace**. 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 Office 365 Message Trace definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Report URL: Enter the URL to use when retrieving report data. Defaults to: `https://reports.office365.com/ecp/reportingwebservice/reporting.svc/MessageTrace`.

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. See About Polling Intervals below.

Authentication Settings

In the **Authentication** section, use the buttons to select an **Authentication method**: Basic, Basic (credentials secret), OAuth, OAuth (text secret), or OAuth Certificate. The default is **OAuth**. All OAuth options rely on the OAuth 2.0 protocol.

Basic Authentication

Selecting **Basic** exposes **Username** and **Password** fields, where you directly enter the HTTP Basic credentials to use on Message Trace API calls.

Basic Secret Authentication

Selecting **Basic (credentials secret)** exposes a **Credentials secret** drop-down, where you select an existing stored secret that references your credentials on the Message Trace API. You can use the adjacent **Create** button to store a new, reusable secret.

OAuth Authentication {#oauth} <!– do they still call Directory ID, Directory ID? ->

The default **OAuth** authentication method exposes the following fields, all required:

- **Client secret**: Directly enter the `client_secret` to pass in the OAuth request parameter.
- **Tenant identifier**: Directory ID (tenant identifier) in Microsoft Entra ID.
- **Client ID**: The `client_id` to pass in the OAuth request parameter.

- **Resource**: Resource parameter to pass in the OAuth request parameter. Defaults to: `https://outlook.office365.com`.

OAuth Secret Authentication

Selecting **OAuth (text secret)** exposes three of the same controls as the default OAuth method, but – as you'd expect – you instead enter the **Client secret** by reference:

- **Client secret**: Use the drop-down to select an existing stored `client_secret` to pass in the OAuth request parameter. You can use the adjacent **Create** button to store a new, reusable secret.
- **Tenant identifier**: Directory ID (tenant identifier) in Microsoft Entra ID.
- **Client ID**: The `client_id` to pass in the OAuth request parameter.
- **Resource**: Resource parameter to pass in the OAuth request parameter. Defaults to: `https://outlook.office365.com`.

OAuth Certificate Authentication

Selecting **OAuth (certificate)** exposes two of the same controls as the above OAuth methods (**Tenant identifier** and **Client ID**). But this authentication method – available in Cribl Stream 4.1.3 and later – foregrounds controls to define the certificate you're using to authenticate. Treat all fields here as required, except for the optional **Passphrase**:

- **Certificate name**: Use the drop-down to select an existing certificate to pass in the OAuth request parameter. You can use the adjacent **Create** button to store a new, reusable cert.
- **Private key path**: Path to the private key to use. The key should be in PEM format. Can reference `$ENV_VARS`.
- **Passphrase**: If your private key requires a passphrase to decrypt it, enter that passphrase here.
- **Certificate path**: Path to the certificate to use. The cert should be in PEM format. Can reference `$ENV_VARS`.
- **Tenant identifier**: Directory ID (tenant identifier) in Microsoft Entra ID.
- **Client ID**: The `client_id` to pass in the OAuth request parameter.
- **Resource**: Resource parameter to pass in the OAuth request parameter. Defaults to: `https://outlook.office365.com`.
- **Subscription Plan**: Select the Office 365 subscription plan for your organization. Options include:
    - **Office 365 Enterprise**
    - **Office 365 GCC**
    - **Office 365 GCC High**
    - **Office 365 DoD**

Optional Settings

Date range start: Backward offset for the head of the search date range. (E.g., `-3h@h`.) Message Trace data is delayed; this parameter (with **Date range end**) compensates for delay and gaps.

Date range end: Backward offset for the tail of the search date range. (E.g., `-2h@h`.) Message Trace data is delayed; this parameter (with **Date range start**) compensates for delay and gaps.

Log level: For data collection's runtime log, set the verbosity level to one of `debug`, `info`, `warn`, or `error`. (If not selected, defaults to `info`.)

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

About Polling Intervals

To poll the Office 365 Message Trace API, Cribl Stream uses the **Poll interval** field's value to establish the cron schedule. (e.g.: `*/${interval} * * * *`).

Because the interval is set in minutes, it must divide evenly into 60 minutes to create a predictable schedule. Dividing 60 by intervals like `1`, `2`, `3`, `4`, `5`, `6`, `10`, `12`, `15`, `20`, or `60` itself yields an integer, so you can enter any of these values.

Cribl Stream will reject intervals like `23`, `42`, or `45`, or `75` – which would yield non-integer results, meaning unpredictable schedules.

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.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Advanced Settings

Keep alive time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `30`.

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`.

Worker timeout (periods): The number of **Keep alive time** periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

Request timeout (secs): Maximum time to wait for an individual Message Trace API request to complete. Defaults to `300` seconds (5 minutes). Enter `0` to disable metering, allowing unlimited response time. Because there is a single request to the Message Trace API per page of data, this timeout is applied at the page (request) level.

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. Defaults to `0`.

Disable time filter: Disables Collector event time filtering when a date range is specified in General Settings. Toggle to `No` to allow filtering.

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 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.

Fields for this Source:

- `__final`
- `__inputId`
- `__isBroken`
- `__source`

How Cribl Stream Pulls Data

The Office 365 Message Trace Source uses a scheduled REST Collector. It runs one collection task every **Poll interval**, and a single Worker will process the collection. The data is paginated, so the Worker might make multiple calls to fetch the data.

Viewing Scheduled Jobs

This Source executes Cribl Stream's scheduled collection jobs. Once you've configured and saved the Source, you can view those jobs' results by reopening the Source's config modal and clicking its **Job Inspector** tab.

Each content type that you enabled gets its own separate scheduled job.

You can also view these jobs (among scheduled jobs for other Collectors and Sources) in the **Monitoring** > **System** > **Job Inspector** > **Currently Scheduled** tab.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a “stuck”, “orphaned,” or “zombie” job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream’s logs.

To keep stuck jobs from running excessively long:

- First, try setting **Advanced** > **Timeout (secs)** to a duration shorter than the default of `600` seconds (10 minutes).
- If adjusting **Timeout (secs)** does not fix the problem, try setting **Advanced** > **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

15.6.3. Office 365 Services

Cribl Stream supports receiving data from the Microsoft Graph service communications API. This facilitates analyzing the status and history of service incidents on multiple Microsoft cloud services, along with associated incident and Message Center communications. For details, see Microsoft's Overview of the Graph API.

Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **YES**

TLS is enabled via the HTTPS protocol on this Source's underlying REST API.

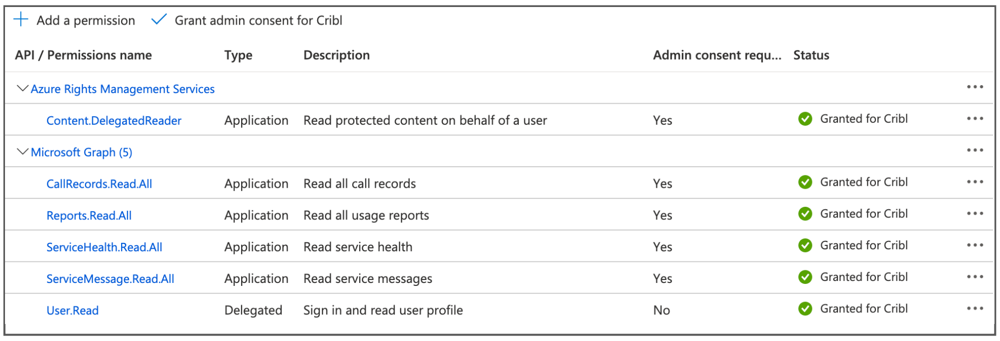
Microsoft has retired its prior Office 365 Service Communications API, forcing a switch to the Graph API mentioned above. Due to a limitation in this new API, Cribl Stream (LogStream) 3.3 and above can no longer collect the **Historical Status** content type that was available in this Source through LogStream 3.2.2.

For more about the Microsoft Graph API, see our Microsoft Graph API Collection guide.

Microsoft Entra ID Permissions

In Microsoft Entra ID, the application representing your Cribl Stream instance must be granted the following permissions to pull data. (The permission **Type** for both must be `Application` – `Delegated` is not sufficient:)

- `ServiceHealth.Read.All`
- `ServiceMessage.Read.All`


+ Add a permission ✓ Grant admin consent for Cribl

API / Permissions name	Type	Description	Admin consent requ...	Status
Azure Rights Management Services				
Content.DelegatedReader	Application	Read protected content on behalf of a user	Yes	✓ Granted for Cribl
Microsoft Graph (5)				
CallRecords.Read.All	Application	Read all call records	Yes	✓ Granted for Cribl
Reports.Read.All	Application	Read all usage reports	Yes	✓ Granted for Cribl
ServiceHealth.Read.All	Application	Read service health	Yes	✓ Granted for Cribl
ServiceMessage.Read.All	Application	Read service messages	Yes	✓ Granted for Cribl
User.Read	Delegated	Sign in and read user profile	No	✓ Granted for Cribl

Registered application permissions

Configuring Cribl Stream to Receive Data from the Service API

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Office 365** > **Services**. 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** >] **Office 365** > **Services**. 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 Office 365 Services definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Tenant ID: Enter the Office 365 Azure tenant ID.

App ID: Enter the Office 365 Azure application ID.

Authentication Settings

Authentication method: Select one of the following buttons.

- **Manual**: This default option provides a **Client secret** field, where you directly enter the required Office 365 Azure client secret.
- **Secret**: This option instead exposes a **Client secret (text secret)** drop-down, from which you select a stored text secret to authenticate with. Click **Create** to configure a new secret.

Optional Settings

Subscription Plan: Select the Office 365 subscription plan for your organization. Options include:

- **Office 365 Enterprise**
- **Office 365 GCC**
- **Office 365 GCC High**
- **Office 365 DoD**

Content Types: See the Content Types section below.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Content Types

Here, you can configure polling separately for the following types of data from the Office 365 Service Communications API:

- **Current Status**: Get a real-time view of current and ongoing service incidents.
- **Messages**: Find incident and Message Center communications.

As of this revision, this Microsoft API provides data for Office 365, Yammer, Dynamics CRM, and Microsoft Intune cloud services. For each of these content types, this section provides the following controls:

Enabled: Toggle this to `Yes` for each service that you want to poll.

Interval: Optionally, override the default polling interval. See About Polling Intervals below.

Log level: Set the verbosity level to one of `debug`, `info` (the default), `warn`, or `error`.

About Polling Intervals

To poll the Office 365 Service Communications API, Cribl Stream uses the **Interval** field's value to establish the search date range and the cron schedule, for example: `*/${interval} * * * *`

Therefore, intervals set in minutes – those for **Current Status** – must divide evenly into 60 minutes to create a predictable schedule. Dividing 60 by intervals like `1`, `2`, `3`, `4`, `5`, `6`, `10`, `12`, `15`, `20`, or `60` itself yields an integer, so you can enter any of these values.

Cribl Stream will reject intervals like `23`, `42`, or `45`, or `75` – which would yield non-integer results, meaning unpredictable schedules.

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.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Advanced Settings

Keep alive time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `30`.

Worker timeout (periods): The number of **Keep alive time** periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

Request timeout (secs): The maximum time period for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `300` (i.e., 5 minutes). Enter `0` to disable timeout metering.

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. Defaults to `0`.

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 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.

Fields for this Source:

- `__final`
- `__inputId`
- `__isBroken`
- `__source`

How Cribl Stream Pulls Data

The Office 365 Services Source retrieves data using Cribl Stream scheduled Collection jobs, which include Discover and Collection phases. The Discover phase task returns the URL of the content to collect.

In the Source's **General Settings** > **Content Types** > **Interval** column, you configure the polling schedule for each Content Type independently.

The job scheduler spreads the Collection tasks across all available Workers. The collected content is paginated, so the collection phase might include multiple calls to fetch data.

Viewing Scheduled Jobs

This Source executes Cribl Stream's scheduled collection jobs. Once you've configured and saved the Source, you can view those jobs' results by reopening the Source's config modal and clicking its **Job Inspector** tab.

Each content type that you enabled gets its own separate scheduled job.

You can also view these jobs (among scheduled jobs for other Collectors and Sources) in the **Monitoring** > **System** > **Job Inspector** > **Currently Scheduled** tab.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a "stuck", "orphaned," or "zombie" job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream's logs.

To keep stuck jobs from running excessively long:

- First, try setting **Advanced** > **Timeout (secs)** to a duration shorter than the default of `300` seconds (5 minutes).
- If adjusting **Timeout (secs)** does not fix the problem, try the global setting **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration. You'll find **Job Timeout** among the task manifest and buffering limits.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.
- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

15.7. Prometheus

15.7.1. Prometheus Scraper

Cribl Stream supports receiving batched data from Prometheus targets. This is a pull Source; to ingest Prometheus streaming data, see Prometheus Remote Write.

> Type: **Pull** | TLS Support: **No** | Event Breaker Support: **No**
>
> This Source assumes that incoming data is snappy-compressed. It does not currently support Prometheus metadata.

Configuring Cribl Stream to Scrape Prometheus Data

> As of v.4.1.2, this Source is deprecated on Cribl Edge. Cribl plans to remove it from Cribl Edge by v.4.2. Please use the Edge-specific Prometheus Edge Scraper Source instead.

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Prometheus** > **Scraper**. 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 **Data** > **Sources** (Stream) or **More** > **Sources** (Edge). From the resulting page's tiles or left nav, select [**Pull** >] **Prometheus** > **Scraper**. Next, click **New Source** to open a **New Source** modal that provides the options below.

General Settings

Additional fields appear in this section depending on what discovery type you select.

Input ID: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Discovery type: Use this drop-down to select a discovery mechanism for targets. See Discovery Type below for the options.

Some **Discovery type** options replace the **Targets** field with additional controls below the **Poll interval** and **Log level** fields – while also adding an **Assume Role** and/or **Target Discovery** left tab to the modal.

Poll interval: Specify how often (in minutes) to scrape targets for metrics. Defaults to `15`. This value must be an integer that divides evenly into `60`.

Log level: Set the verbosity level to one of `debug`, `info` (the default), `warn`, or `error`.

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.

Values can be in URL or `host[:port]` format, e.g.: `http://localhost:9090/metrics`, `localhost:9090`, or `localhost`.

If you specify only `host[:port]`, the endpoint will resolve to: `http://host[:port]/metrics` – that is, the protocol will be prepended and the `/metrics` path will be appended in all cases, even if you add a path. If you need a nonstandard path, you must enter the full URL (including protocol) with path. For example: `http://localhost:9090/my/custom/path`.

DNS Discovery

The `DNS` option adds a Target Discovery tab to the modal, and adds two extra fields to its **General Settings** tab:

- **DNS names**: Enter a list of DNS names to resolve.
- **Record type**: Select the DNS record type to resolve. Defaults to `SRV` (Service). Other options are `A` or `AAAA`.

AWS EC2 Discovery

The `AWS EC2` option adds Assume Role and Target Discovery tabs to the modal, and adds one extra field to the **Optional Settings** tab:

- **Region**: Select the AWS region in which to discover EC2 instances with metrics endpoints to scrape.

This option also adds controls in the **Advanced Settings** tab, as described below.

Optional Settings

Extra dimensions: Specify the dimensions to include in events. Defaults to `host` and `source`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication (Prometheus)

Use the **Authentication method** drop-down to select one of these authentication options for Prometheus:

- **Manual**: In the resulting **Username** and **Password** fields, enter Basic authentication credentials corresponding to your Prometheus targets.
- **Secret**: This option exposes a **Secret** drop-down, in which you can select a stored secret that references your credentials described above. The secret can reside in Cribl Stream's internal secrets manager or (if enabled) in an external KMS. Click **Create** if you need to configure a new secret.

Assume Role

With the AWS EC2 target discovery type, you can configure AssumeRole behavior on AWS.

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

- **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.
- **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).

Target Discovery

Setting the General Settings > Discovery type drop-down to DNS or AWS EC2 exposes this tab. These two discovery types expose different controls here.

Target Discovery for DNS

Setting the Discovery type drop-down to DNS exposes the following **Target Discovery** fields.

Metrics protocol: Select `http` (the default) or `https` as the protocol to use when collecting metrics.

Metrics path: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.

Target Discovery for AWS

Setting the Discovery type drop-down to AWS EC2 exposes the following **Target Discovery** controls. The first controls is a special case:

- **Authentication method**: Select the **Auto**, **Manual**, or **Secret** button to determine how Cribl Stream will authenticate against AWS. Each selection changes the fields displayed on this tab – see AWS Authentication Options for details.

These remaining controls are displayed for all **Authentication method** selections:

- **Metrics protocol**: Select `http` (the default) or `https` as the protocol to use when collecting metrics.
- **Metrics port**: Specify the port number to append to the metrics URL for discovered targets. Defaults to `9090`.
- **Metrics path**: Specify a path to use when collecting metrics from discovered targets. Defaults to `/metrics`.
- **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.)

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 Cribl Stream 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. This option displays the same fields as Auto, plus:

- **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 the same fields as Auto, plus:

- **Secret key pair**: Use the drop-down to select a secret key pair that you've configured in Cribl Stream'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.

Advanced Settings

Keep alive time (seconds): How often workers should check in with the scheduler to keep job subscription alive. Defaults to `60` seconds.

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`.

Worker timeout (periods): How many **Keep alive time** periods before an inactive worker's job subscription will be revoked. Defaults to `3` periods.

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 two additional settings appear only when **Optional Settings** > **Discovery Type** is set to `AWS EC2`.

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.

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.

Fields for this Source:

- `__source`
- `__isBroken`
- `__inputId`
- `__final`
- `__criblMetrics`
- `__channel`
- `__cloneCount`

How Cribl Stream Pulls Data

The Prometheus Source retrieves data using Cribl Stream scheduled Collection jobs. You determine the schedule using your **Poll interval** entry.

With the `DNS` or `AWS EC2` **Discovery Type**, these jobs include both Discover and Collection phases. The Discover phase runs on a single Worker, and returns 1 collection task per discovered target.

The job scheduler spreads the Collection tasks across all available Workers.

Viewing Scheduled Jobs

This Source executes Cribl Stream's scheduled collection jobs. Once you've configured and saved the Source, you can view those jobs' results by reopening the Source's config modal and clicking its **Job Inspector** tab.

Each content type that you enabled gets its own separate scheduled job.

You can also view these jobs (among scheduled jobs for other Collectors and Sources) in the **Monitoring** > **System** > **Job Inspector** > **Currently Scheduled** tab.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

15.7.2. Prometheus Remote Write

Cribl Stream 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 Stream to Receive Metrics from Prometheus Remote Write Sources

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If

the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data.

Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.

- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__criblMetrics`
- `__final`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`

- `__srcIpPort` – See details below.
- `_time`
- `_value`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Prometheus Remote Write client sending data to this Source.

When any proxies (including load balancers) lie between the Prometheus Remote Write client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **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 Stream 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 Stream supports the `timer` type in addition to `counter` and `gauge`.

15.7.3. Grafana

Cribl Stream 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 Stream to Receive Metrics and Logs from Grafana Agent Sources

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream 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.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.

- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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 Stream 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 Stream supports the `timer` type in addition to `counter` and `gauge`.

15.7.4. Loki

Cribl Stream 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 Stream to Receive Loki Logs Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 Stream:

- **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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If

the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data.

Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream 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.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__final`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`
- `__labels` – Will contain all the labels found in each event's corresponding Loki stream.
- `__srcIpPort` – See details below.
- `_time`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Loki client sending data to this Source.

When any proxies (including load balancers) lie between the Loki client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **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.)

15.8. Splunk

15.8.1. Splunk HEC

Cribl Stream 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 Stream to Receive Data over Splunk HEC

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Address: Enter the hostname/IP on which to listen for HTTP(S) data. Such as: `localhost` or `0.0.0.0`. Supports IPv4 and IPv6 addresses.

Port: Enter the port number.

Splunk HEC endpoint: Absolute path on which to listen for the Splunk HTTP Event Collector API requests. Defaults to `/services/collector`.

This single endpoint supports JSON events via `/event`, health checks via `/health`, raw events via `/raw`, and Splunk S2S events via `/s2s`. See the Splunk REST API endpoints documentation and the examples below. The Source will automatically detect where to forward the request.

Optional Settings

Allowed Indexes: List the values allowed in the HEC event index field. Allows wildcards. Leave blank to skip validation.

Splunk HEC acks: Whether to enable Splunk HEC acknowledgments. Defaults to `No`. See Working with HEC Acks below to learn about context and limitations.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Working with HEC Acks

Cribl Stream sends a `200` or similar HTTP response code when it receives an event.

Some senders also send **ack requests**, which differ from events. This type of request asks Cribl Stream to acknowledge delivery of an array of event IDs. (A sender might **require** HEC acks to be enabled. Cribl does not maintain a comprehensive list of senders that require acks – please refer to your sender's documentation.)

How Cribl Stream responds depends on **Optional Settings** > **Splunk HEC acks**:

- When **Splunk HEC acks** is turned on, Cribl Stream will respond to an ack request with an acknowledgement affirming that Cribl Stream received each of the events whose IDs were listed in the array.
- When **Splunk HEC acks** is turned off, Cribl Stream will respond to an ack request with a `400` error and text saying that ACK is disabled.

It makes sense to turn **Splunk HEC acks** on for senders that keep TCP connections open while waiting for an ack, because this behavior can exhaust available file descriptors.

There is a caveat to how HEC acks work in Cribl Stream: The ack response simply cites whatever event IDs appeared in the ack request, regardless of whether or not those events were ever received or processed by Cribl Stream. In that sense, what Cribl Stream sends is a "fake ack:" unlike acknowledgements in some other systems, the ack in itself is meaningless. It mainly serves to spare a sender from keeping TCP connections open needlessly.

To determine whether Cribl Stream successfully ingested an event, pay no attention to acks. Instead, verify that the sender received a `200` or similar HTTP response from the Splunk HEC source.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as, the system's CA). Defaults to `No`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

> You can optimize Workers' startup connections and CPU load at **Group 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**.

Enable token: Controls the status of the specified authentication token. When set to `Yes`, the specified auth token is active and can be used for client access. If it's set to `No`, the token is disabled, causing it to be skipped and ignored during initialization. This means that any client attempting to use a disabled token will not be granted access.

Disabling a token can be helpful when you need to temporarily stop receiving data from a particular service or application. Disabling a token prevents the service or application from sending data to your endpoint without having to completely delete the token.

See also Periodic Logging for information on how auth tokens affect product logging.

Description: Optional 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 Stream 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 Stream 4.1.2 and later.

CORS allowed headers: List HTTP headers that Cribl Stream 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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 (that is, the "actual" as opposed to preflight request). The Splunk HEC Source can then respond with an `Access-Control-Allow-Headers` header **if and only if** it's also sending the `Access-Control-Allow-Origin` header.

CORS allowed headers value	What Splunk HEC Source does	Does Browser then allow code to read response?
`*`	Send `Access-Control-Allow-Headers` header with the same list the client sent.	Yes
Multiple wildcard values	If any headers from the client's list match wildcards, send `Access-Control-Allow-Headers` header listing the matching headers.	Yes
Nothing (not configured)	Do not send an `Access-Control-Allow-`	Probably not

CORS	`Headers` header.	Does Browser then

Format and Endpoint Examples

Splunk HEC to Cribl Stream

- Configure Cribl Stream to listen on port `8088` with an auth token of `myToken42`.
- Send a payload to your Cribl Stream receiver.

Note: Token specification can be either `Splunk <token>` or `<token>`.

Splunk HEC - JSON Event Examples　Splunk HEC - Raw Event Example

Splunk HEC - Health Event Example

```
curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization: m

curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken

# Multiple Events
curl -k http://<myCriblHost>:8088/services/collector -H 'Authorization: myToken

# Metrics Events
curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization: m

curl -k http://<myCriblHost>:8088/services/collector/event -H 'Authorization: m

# Send the auth token as a query parameter, with no additional configuration
curl -k "http://<myCriblHost>:8088/services/collector/event?token=mToken42" -d
```

Splunk HEC to Cribl.Cloud

- Navigate to Cribl.Cloud's Splunk HEC Source > **Auth Tokens** tab.
- Copy your token out of the **Token** field.
- From the command line, use `https`, your Cribl.Cloud portal's **Ingest Endpoint** and port, and the token's value:

Splunk HEC > Cribl Cloud endpoint

```
curl -k "https://default.main-<Your-Org-ID>.cribl.cloud:8088/services/collector" \
    -H "Authorization: <token_value>" \
    -d '{"event": "Goats are better than ponies."}{"event": "Goats are better climb
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Worker Groups.

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

If one or more auth tokens are configured and enabled, Cribl Stream logs requests and events for each enabled auth token individually. Since the tokens themselves are redacted for security, Cribl Stream logs the initial text of the token description to help you identify which token a given log is for.

If no auth token is configured and enabled, Cribl Stream simply logs overall statistics about incoming requests and ingested events.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

15.8.2. Splunk Search

Cribl Stream supports receiving Splunk search data from Splunk Search.

Type: **Pull** | TLS Support: **Yes** | Event Breaker Support: **YES**

Configuring Cribl Stream to Receive Splunk Search Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **Splunk Search**. 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** >] **Splunk Search**. 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 Splunk Search Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Cron schedule: Enter a cron expression to define the schedule on which to run this job. Defaults to one run every 15 minutes. The **Estimated Schedule** below this field shows the next few collection runs, as examples of the cron interval you've scheduled.

You enter the **Cron schedule** expression in UTC time, but the **Estimated Schedule** examples are displayed in local time.

Search Settings

Search: Enter the Splunk query. For example: `index=myAppLogs level=error channel=myApp OR | mstats avg(myStat) as myStat WHERE index=myStatsIndex`.

Search head: Enter the search head base URL. The default is `https://localhost:8089`.

Earliest: You can enter the earliest time boundary for the search. This maybe be an exact or relative time. For example: `2022-01-14T12:00:00Z` or `-16m@m`.

Latest: You can enter the latest time boundary for the search. This maybe be an exact or relative time. For example: `2022-01-14T12:00:00Z` or `-16m@m`.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

In the **Authentication type** drop-down, select one of these options:

- **None**: Don't use authentication. Compatible with REST servers like AWS, where you embed a secret directly in the request URL.
- **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.
- **Bearer Token**: Provide the `token` value configured and generated in Splunk.
- **Bearer Token (text secret)**: Provide the Bearer Token referenced by a secret. Select a stored text secret in the resulting **Token (text secret)** drop-down, or click **Create** to configure a new secret.

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 the `Splunk Search Ruleset`.

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.

Retries

Retry type: The algorithm to use when performing HTTP retries. Options include `Backoff` (the default), `Static`, and `Disabled`.

Initial retry interval (ms): Time interval between failed request and first retry (kickoff). Maximum allowed value is 20,000 ms (1/3 minute). A value of `0` means retry immediately until reaching the retry limit in **Max retries**.

Max retries: Maximum number of times to retry a failed HTTP request. Defaults to `5`. Maximum: `20`. A value of `0` means don't retry at all.

Backoff multiplier: Base for exponential backoff. A value of `2` (default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, etc.

Retry HTTP codes: List of HTTP codes that trigger a retry. Leave empty to use the defaults (`429` and `503`). Cribl Stream does not retry codes in the `200` series.

Honor Retry-After header: When toggled to `Yes` (the default) and the `retry-after` header is present, Cribl Stream honors any `retry-after` header that specifies a delay, up to a maximum of 20 seconds. Cribl Stream always ignores `retry-after` headers that specify a delay longer than 20 seconds.

Cribl Stream will log a warning message with the delay value retrieved from the `retry-after` header (converted to ms).

When toggled to `No`, Cribl Stream ignores all `retry-after` headers.

Advanced Settings

Search endpoint: Rest API used to conduct a search. Defaults to `services/search/jobs/export`.

Output mode: Format of the returned output. Defaults to JSON format.To parse the returned JSON, add the Cribl event breaker which parses newline delimited events in the Event Breakers tab.

Events returned from Splunk search can also be returned in the more compact CSV format. To use CSV format, set the **Output mode** to CSV and specify the CSV event breaker in the Event Breakers tab.

Endpoint parameters: Optional HTTP request parameters to append to the request URL. These refine or narrow the request. Click **Add Parameter** to add parameters as key-value pairs:

- **Name**: Field name.
- **Value**: JavaScript expression to compute the field's value (can be a constant).

Endpoint headers:: Click **Add Header** to (optionally) add request headers to send to the endpoint, as key-value pairs:

- **Name**: Header name.
- **Value**: JavaScript expression to compute the header's value, normally enclosed in backticks (e.g., `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (e.g., `earliest`) are evaluated as strings.

Log level: Set the verbosity level for the data collection's runtime log.

Keep Alive Time (Seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `30`.

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`.

Worker timeout (periods): The number of Keep Alive Time periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

Request Timeout (secs): Here, you can set a maximum time period (in seconds) for an HTTP request to complete before Cribl Stream treats it as timed out. Defaults to `0`, which disables timeout metering.

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.

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.

Fields for this Source:

- `__inputId`
- `__outputMode`

How Cribl Stream Pulls Data

This Collector-based Source will gather data from the specified **Search head** URL repeatedly, on the interval specified in the **Cron schedule** field. A single Worker executes each collection job.

If the Leader goes down, search jobs in progress will complete, but future scheduled searches will not run until the Leader relaunches.

Mitigating Stuck-Job Problems

Occasionally, a scheduled job fails, but continues running for hours or even days, until someone intervenes and cancels it. If left alone, such a "stuck", "orphaned," or "zombie" job will never complete. This can cause missing events in downstream receivers, along with `HTTP timeout` or similar errors in Cribl Stream's logs.

To keep stuck jobs from running excessively long:

- First, try setting **Advanced** > **Request Timeout (secs)** – whose default of `0` means "wait forever" – to a desired maximum duration.
- If adjusting **Timeout (secs)** does not fix the problem, try setting **Advanced** > **Job Timeout** – whose default of `0` allows a job to run indefinitely – to a desired maximum duration.

Using these settings in tandem works like this:

- **Timeout (secs)** limits the time that Cribl Stream will wait for an HTTP request to complete.

- Then, if a job gets stuck and keeps running beyond that limit, **Job Timeout** can catch and terminate the job, because it monitors the overall time the job has been running.

15.8.3. Splunk TCP

Cribl Stream supports receiving Splunk data from Universal or Heavy Forwarders.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Stream to Receive Splunk TCP Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.

> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> As of Cribl Stream 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 **Group 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.

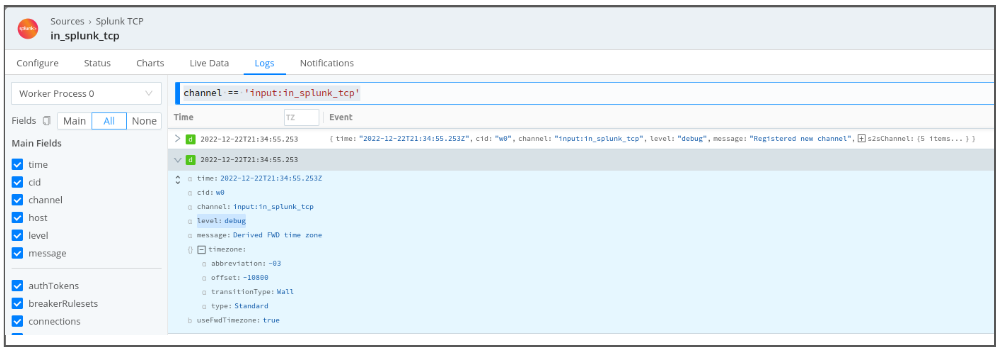
Using the UF Time Zone

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.
```

This gap can be problematic, especially if the originating Universal Forwarder is in a different time zone from the processing Worker 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 Stream's own logs as shown below:



```
channel == 'input:in_splunk_tcp'
```

```
2022-12-22T21:34:55.253   { time: "2022-12-22T21:34:55.253Z", cid: "w0", channel: "input:in_splunk_tcp", level: "debug", message: "Registered new channel", s2sChannel: {5 items... } }
2022-12-22T21:34:55.253
  time: 2022-12-22T21:34:55.253Z
  cid: w0
  channel: input:in_splunk_tcp
  level: debug
  message: Derived FWD time zone
  timezone:
    abbreviation: -03
    offset: -10800
    transitionType: Wall
    type: Standard
  useFwdTimezone: true
```

Time zone information in logs

Setting the Log Level for Connection Messages

When a Splunk forwarder connects to Cribl Stream, Cribl Stream logs the following message at the `debug` level: `Connection with forwarder has been established successfully.`

To see this message, set the `:forwarders` level to `debug`.

Each message contains details specific to the forwarder, such as the protocol, Splunk version, or remote host, to name a few.

In some situations, logging each incoming connection can produce many messages, which can make it hard to find other messages.

You can adjust the level of these connection messages. To do so, follow these steps:

1. Select **Settings** > **Global Settings** > **Logging** > **Levels**.
2. Search for the channel that logs the connection messages. It will have a name in the form `input:<source-id>:forwarders`. For example: `input:in_splunk_tcp:forwarders`.
3. Set the channel to the log level you prefer, such as `debug` or `silly`.
4. Click **Save** to save your setting.
5. Commit and deploy the change.

Internal Fields

Cribl 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.

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>, ...
enableOldS2SProtocol = true

[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ..
compressed = false
sendCookedData = true
# As of Splunk 6.5, using forceTimebasedAutoLB is no longer recommended. Ensure
# forceTimebasedAutoLB = false
```

If your use case requires compression, use SSL forwarding to compress the data stream.

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Worker Groups.

Preventing Data Loss with v3

If you set **Max S2S version** to `v3` and are using Splunk 9.1.0 or later, Cribl recommends that you use the `enableOldS2SProtocol = true` setting shown above to avoid data loss. If you are working with `v3` and a Splunk version earlier than 9.1.0, you should use `negotiateProtocolLevel = 0`. Depending on your environment, enabling `negotiateProtocolLevel` with a non-`0` value could cause Cribl Stream to not accept data from the forwarder.

If you set **Max S2S version** to `v4`, these settings are not necessary. The Splunk receiver will detect which version is in use and automatically use the correct handler.

> See Internal Fields for information on the `__s2sVersion` field.

Troubleshooting 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.

15.9. Internal

15.9.1. Cribl Internal

The Cribl Internal Source enables you to capture and send Cribl Stream's own internal **logs** and **metrics** through Routes and Pipelines.

Type: **Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

Scope and Purpose

In distributed mode, this Source's CriblLogs option can process internal logs only from Worker Processes. (Logs on the Leader remain on the Leader, because the Leader Node is not part of any processing path.)

In both distributed and single-instance mode, this Source's CriblLogs option omits API Process logs, meaning that it omits telemetry/license-validation traffic. You can, however, use a Script Collector to check for API Server (or Worker Group) events.

This Source's CriblMetrics option offers Cribl's most up-to-date and precise view of event throughput and transformation, aggregated every 2 seconds at the Worker Process level. The CriblLogs option is less granular – tracking logs that are written once per minute – so CriblLogs might fail to reflect Worker Process crashes or restarts.

In Cribl.Cloud, the CriblLogs internal Source is only available on hybrid, customer-managed Worker Nodes.

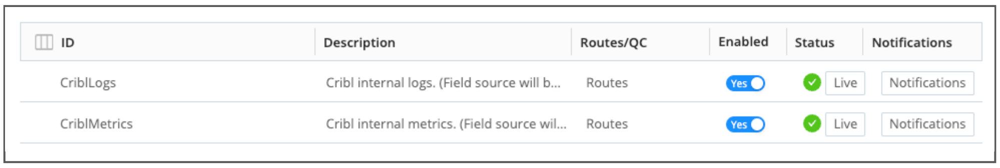
Configuring Cribl Internal Logs/Metrics as a Data Source

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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, toggle **Enabled** to `Yes`. Confirm your choice in the resulting message box. The screenshot below shows both options successfully enabled.



ID	Description	Routes/QC	Enabled	Status	Notifications
CriblLogs	Cribl internal logs. (Field source will b...	Routes	Yes	✓ Live	Notifications
CriblMetrics	Cribl internal metrics. (Field source wil...	Routes	Yes	✓ Live	Notifications

Cribl Internal Sources – click either or both of these rows to configure

CriblLogs – General Settings

Enabled: This duplicates the parent page's **Enabled** toggle. Keep it at `Yes` to enable Cribl logs as a Source.

Input ID: Enter a unique name to identify this CriblLogs Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

CriblLogs – Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

CriblMetrics – General Settings

Enabled: This duplicates the parent page's **Enabled** toggle. Keep it at `Yes` to enable Cribl metrics as a Source.

Input ID: Enter a unique name to identify this CriblMetrics Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

CriblMetrics – Optional Settings

Metric name prefix: Enter an optional prefix that will be applied to metrics provided by Cribl Stream. The prefix defaults to `cribl.logstream`.

If Cribl Stream 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 Stream 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 this Pipeline to your **Cribl Internal** Source as a pre-processing Pipeline. The Pipeline's **Rollup Metrics** Function has a default **Time Window** of 30 seconds, which you can adjust to a different granularity as needed. This provides a second lever to reduce granularity, in addition to the Full fidelity toggle described above.

Omitting `sourcetype`

You can easily drop the `sourcetype` attribute from metrics events, leaving only `event_sourcetype`. This will prevent duplicate `sourcetype` events from being routed to Destinations.

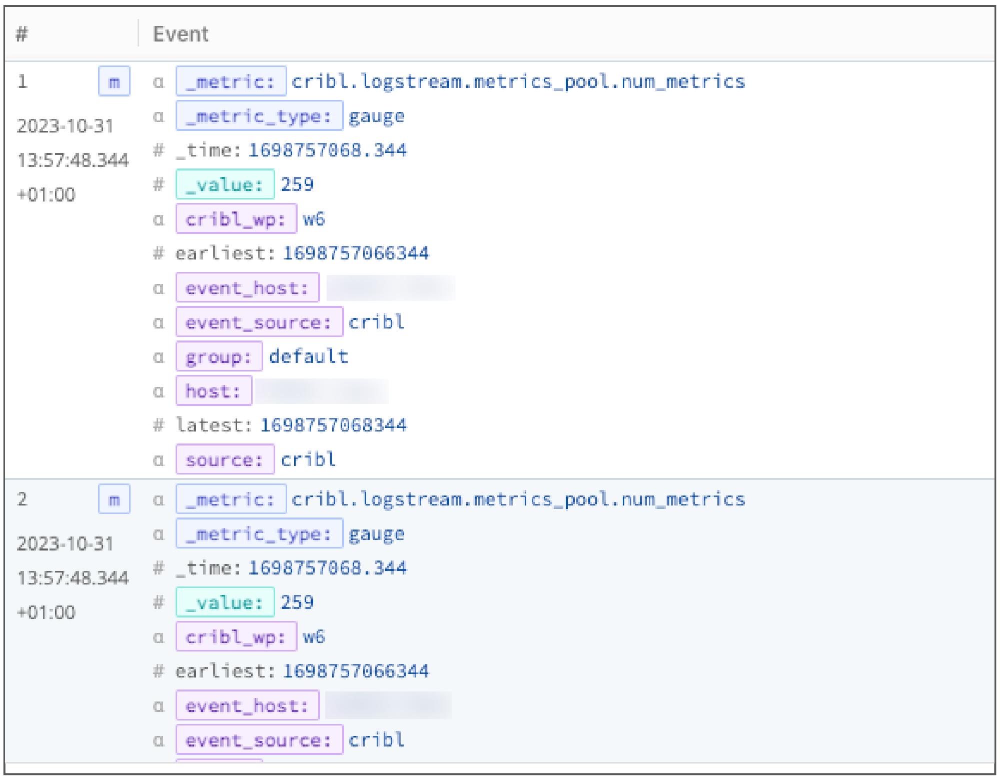
To do this: In the same `cribl_metrics_rollup` pre-processing Pipeline (or a clone) that you attach to your Source, enable the final Eval Function, which applies this **Filter** expression to remove the `sourcetype` field: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

Duplicated Fields/Dimensions

The CriblMetrics Source operates on metrics that Worker 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:



```
#   Event
1                  m   α  _metric: cribl.logstream.metrics_pool.num_metrics
2023-10-31             α  _metric_type: gauge
13:57:48.344           #  _time: 1698757068.344
+01:00                 #  _value: 259
                       α  cribl_wp: w6
                       #  earliest: 1698757066344
                       α  event_host:
                       α  event_source: cribl
                       α  group: default
                       α  host:
                       #  latest: 1698757068344
                       α  source: cribl
2                  m   α  _metric: cribl.logstream.metrics_pool.num_metrics
2023-10-31             α  _metric_type: gauge
13:57:48.344           #  _time: 1698757068.344
+01:00                 #  _value: 259
                       α  cribl_wp: w6
                       #  earliest: 1698757066344
                       α  event_host:
                       α  event_source: cribl
```

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.

15.9.2. Cribl HTTP

The Cribl HTTP Source receives data from a Cribl HTTP Destination in the same Distributed deployment, including Cribl.Cloud, at no charge. It's common to send data from Edge and Stream within the same deployment using a Cribl HTTP Destination and Cribl HTTP Source pair.

The Cribl HTTP Source is available only in Distributed deployments. In single-instance mode or for testing, you can substitute it with the Raw HTTP/S Source. However, this substitution will not facilitate sending all internal fields, as described below.

Type: **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 TCP traffic.

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 on another Worker, you're billed for ingress only once – when Cribl first receives the data. All data subsequently relayed to other Workers via a Cribl HTTP Destination/Source pair is not charged.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (on-prem) Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl HTTP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Edge to on-prem Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.
- Create an on-prem Cribl HTTP Destination.
- Create a Cribl HTTP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl HTTP Destination, and from there to the Cribl HTTP Source in Cribl.Cloud.
- On Cribl-managed Cribl.Cloud 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 Node that is connected to the same Leader as the Cribl HTTP Source(s).
- The Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).
- Cribl 3.5.4 was a breakpoint in Cribl HTTP Leader/Worker communications. Nodes running the Cribl HTTP Source on Cribl 3.5.4 and later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl HTTP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

Configuring the Cribl HTTP Source

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 the Cribl Stream UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

TLS Settings (Server Side)

Enabled Defaults to `No`. When toggled to `Yes`, 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 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.

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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.

- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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.)

15.9.3. Cribl TCP

The Cribl TCP Source receives data from a Cribl TCP Destination in the same Distributed deployment, including Cribl.Cloud, at no charge. It's common to send data from Edge and Stream within the same deployment using a Cribl TCP Destination and Cribl TCP Source pair.

The Cribl TCP Source is available only in Distributed deployments. In single-instance mode or for testing, you can substitute it with the TCP JSON Source. However, this substitution will not facilitate sending all internal fields, as described below.

Type: **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 TCP traffic.

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 on another Worker, you're billed for ingress only once – when Cribl first receives the data. All data subsequently transferred to other Workers via a Cribl TCP Destination/Source pair is not charged.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (on-prem) 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 Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.
- Create an on-prem Cribl TCP Destination.
- Create a Cribl TCP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node, configure a File System Collector to send data to the Cribl TCP Destination, and from there to the Cribl TCP Source in Cribl.Cloud.
- On Cribl-managed Cribl.Cloud 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 Node that is connected to the same Leader as the Cribl TCP Source.
- When you configure the Cribl TCP Destination, its **Address** and **Port** values must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Cribl 3.5.4 was a breakpoint in Cribl TCP Leader/Worker communications. Nodes running the Cribl TCP Source on Cribl 3.5.4 and later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl TCP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

Configuring the Cribl TCP Source

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 the Cribl Stream UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (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 Stream 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 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.

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 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.

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.

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

This kind of periodic logging helps you determine whether a Source is in fact still healthy even when no data is coming in.

15.9.4. Cribl Stream (Deprecated)

This Source was deprecated as of Cribl Stream 3.5, and has been removed as of v.4.0. Please instead use the Cribl TCP or the Cribl HTTP Source to enable Worker 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 Worker 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 Worker Node that is connected to the same Leader. The following instructions cover configuring such a TCP JSON Destination.

Type: **Internal** | TLS Support: **YES** | Event Breaker Support: **No**

Use New Sources Instead

The replacement Sources, Cribl TCP and the Cribl HTTP, don't rely on IP filtering, for the following reasons:

- Load balancers and/or proxies between the Cribl Destination and Cribl Source can change the IP address, resulting in a bad match and rejected ingest.
- A Lookup table of all IP addresses needed to be sent to each Worker Node/Edge Node from the Leader, which is not scalable.
- The Lookup table of IP addresses required constant communication between the Worker Node/Edge Nodes and the Leader, making this fragile and placing an arbitrary reliance on the Leader that shouldn't be there.

How It Works

You can use the Cribl Stream Source to send data between Workers. In a hybrid Cribl.Cloud deployment, using this Source ensures that you're billed for ingress only once – when the data is originally received. All other data transferred into Workers via the Cribl Stream Source is not charged.

This use case is common in deployments where 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 the Cribl Stream UI. Use a tab or hard return between (arbitrary) tag names. These tags aren't added to processed events.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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`

15.10. System

15.10.1. Datagen

Cribl Stream supports generating data from datagen files, as detailed in Using Datagens. When a datagen is enabled, each Worker Process uses the specified data generator file to generate events. These events proceed through Routes and Pipelines, or through a QuickConnect configuration, to configured Destinations. Whichever Worker Process generated an event from the file will also send the same event.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

Configuring Cribl Stream to Generate Sample Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 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.

Fields for this Source:

- `__inputId`

15.10.2. Exec

The Exec Source enables you to periodically execute a command and collect its `stdout` output. This is typically used in cases when Cribl Stream cannot accomplish collection with native Collectors or other Sources.

Available in Cribl.Cloud: **No** | Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

This Source allows you to run **almost anything** on the host system. Make sure you understand the security and other impacts of commands you plan to execute, before proceeding.

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 **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Command: Command to execute; supports Bourne shell syntax.

Schedule type: Use the buttons to select either `Interval` or `Cron`. 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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.

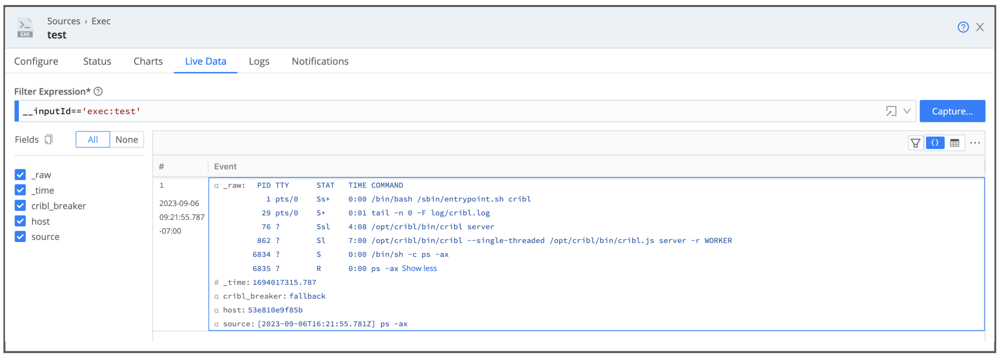
Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl Stream uses internal fields to assist in forwarding data to a Destination.

You can find the following event fields on the **Live Data** tab of the selected Exec Source, including the hostname and source of the event:

- `_raw`
- `_time`
- `cribl_breaker`
- `host`
- `source`



Filter Expression*

```
__inputId=='exec:test'
```

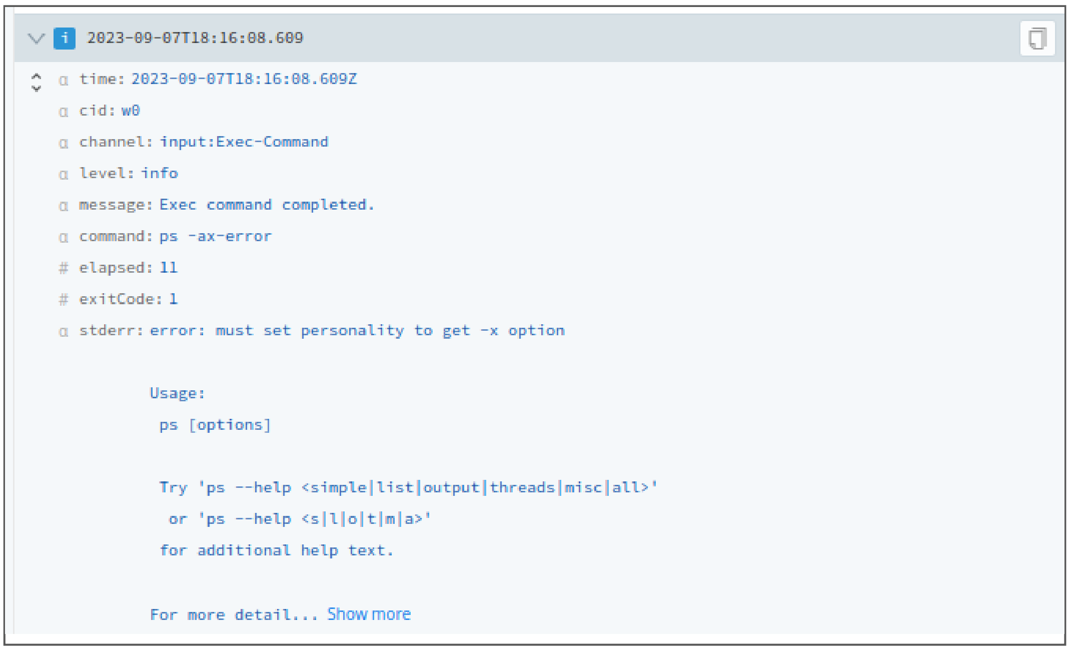
- ✓ _raw
- ✓ _time
- ✓ cribl_breaker
- ✓ host
- ✓ source

```
#   Event
1   _raw:  PID TTY      STAT   TIME COMMAND
             1 pts/0    Ss+    0:00 /bin/bash /sbin/entrypoint.sh cribl
            29 pts/0    S+     0:01 tail -n 0 -F log/cribl.log
            76 ?        Ssl    4:08 /opt/cribl/bin/cribl server
           862 ?        Sl     7:00 /opt/cribl/bin/cribl --single-threaded /opt/cribl/bin/cribl.js server -r WORKER
          6834 ?        S      0:00 /bin/sh -c ps -ax
          6835 ?        R      0:00 ps -ax Show less
2023-09-06  _time: 1694017315.787
09:21:55.787  cribl_breaker: fallback
-07:00  host: 53e810e9f85b
    source: [2023-09-06T16:21:55.781Z] ps -ax
```

Event fields

Troubleshooting

If a command isn't running as expected, the **Logs** tab can help you identify issues by displaying the command executed, its elapsed time, and its exit code.



```
2023-09-07T18:16:08.609
time: 2023-09-07T18:16:08.609Z
cid: w0
channel: input:Exec-Command
level: info
message: Exec command completed.
command: ps -ax-error
elapsed: 11
exitCode: 1
stderr: error: must set personality to get -x option

        Usage:
         ps [options]

         Try 'ps --help <simple|list|output|threads|misc|all>'
          or 'ps --help <s|l|o|t|m|a>'
         for additional help text.

        For more detail... Show more
```

Event fields

15.10.3. 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.

Available in Cribl.Cloud: **No** | Type: **System** | TLS Support: **N/A** | Event Breaker Support: **Yes**

As of v.4.2.x, this Source can process the following file formats:

- **Compressed/archived**: `zip`, `gzip`, `zstd`, and `tar`. Cribl Stream 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.

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 **Worker Group** to configure. Next, you have two options:

Configure via QuickConnect

1. To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge).
2. Click **Add Source** at left. From the resulting drawer's tiles, select **System and Internal** > **File Monitor**.
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 **System and Internal** > **File Monitor**.
3. Click **Add Source** to open the **New Source** modal.

General Settings

Enabled: Toggle to `Yes` to enable the Source.

Input ID: Enter a unique name to identify this File Monitor Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 Stream'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 Stream'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.

Force text format: If toggled to `Yes`, the Source displays ingested data as text in the event (when the data isn't in a compressed or archived file). This option is helpful if you have files that contain binary mixed with text where the data should be streamed as all text (for example, AuditD logs can contain this type of data).

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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).

Hash length: How many file header bytes to use in a hash for identifying a file uniquely. Defaults to `256` bytes. For details, see Configuring Hash Lengths.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

This Source defaults to QuickConnect.

Internal Fields

Cribl 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.

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 Stream is stopped, files are rotated, and Cribl Stream 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.

Configuring Hash Lengths

For files with identical first lines, configure the hash length to be longer than the first line to ensure that the header includes something unique. It is common for a batch of `CSV` files or `IIS` logs to have identical first lines listing columns that appear in subsequent lines.

If you don't adjust the hash length, all files with headers larger than `256 bytes` will appear to File Monitor as one file. As long as the hash length is greater than the header length, and the subsequent lines include something unique, the File Monitor Source will identify them as different files.

Using Allowlists Effectively

Done properly, file monitoring collects exactly the files you need to see, without duplication. You want to avoid inadvertently creating overlapping monitor groups that deliver duplicates, because this can "run the meter" twice for affected files, potentially causing license issues.

The key is knowing how to craft allowlists.

Crafting Allowlists: Principles

To get the results you want from file monitoring, always apply the following principles.

Write allowlists to explicitly include the full path of files you want to collect.

The search path itself is **not** excluded from the match. For example, if you want to collect the file `/var/log/messages` when you've set your search path to `/var/log`, your filename allowlist needs to be either `/var/log/messages`, `/var/*/messages`, or `*/log/messages`.

Exclusions (`!`) also need to address the full file path. For example, if you want to collect everything from `/var/log/*` **except for** `/var/log/apache/*`, your allowlist must include one of the following:

- `!/var/log/apache/*`
- `!*/apache/*`
- `!*/log/apache/*`

Wildcards (`*`) match **any** part of the file path:

- `*log` matches `/var/foo/bar/myfile.log`.
- `*.log` matches anything ending in `.log`.
- `/var/log/*` matches anything in `/var/log`, subject to the **Max Depth** setting.

Put explicit matches (and exclusions) at the beginning of the allowlist, ahead of wildcard matches.

For example, an allowlist of `/var/log/*` followed by `!/var/log/apache/*` will fail to exclude the file `/var/log/apache/foo` because that file matches the first allowlist entry.

In allowlists, order matters!

Be as specific as possible, whenever possible, to prevent accidental matches.

Each of the following examples shows how to avoid matching unwanted files within the directories that you're monitoring.

To collect `/var/log/boot.log` (and older versions of `boot.log.*`) as well as `/var/log/messages`:

Search path	Filename allowlist
`/var/log`	`/var/log/boot.log`, `/var/log/messages`

To collect everything in `/var/log` while excluding `/var/log/apache/*`:

Search path	Filename allowlist
`/var/log`	`!/var/log/apache/*`, `/var/log/*`

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!

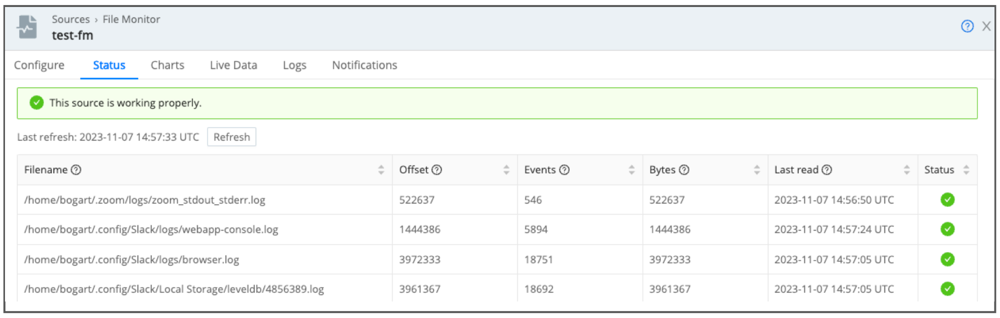
Here's the example:

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 Stream'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:



This source is working properly.

Last refresh: 2023-11-07 14:57:33 UTC

Filename	Offset	Events	Bytes	Last read	Status
/home/bogart/.zoom/logs/zoom_stdout_stderr.log	522637	546	522637	2023-11-07 14:56:50 UTC	✓
/home/bogart/.config/Slack/logs/webapp-console.log	1444386	5894	1444386	2023-11-07 14:57:24 UTC	✓
/home/bogart/.config/Slack/logs/browser.log	3972333	18751	3972333	2023-11-07 14:57:05 UTC	✓
/home/bogart/.config/Slack/Local Storage/leveldb/4856389.log	3961367	18692	3961367	2023-11-07 14:57:05 UTC	✓

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`	`*`

15.10.4. 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** | TLS Support: **N/A** | Event Breaker Support: **No**

The Journal Files Source is available on Cribl.Cloud hybrid Workers, but not on Cribl-managed Cribl.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 Stream ships with a single, disabled instance of the Journal Files Source with a default configuration. The search path is configured to point to `$CRIBL_EDGE_FS_ROOT/var/log/journal/$MACHINE_ID` and defaults to streaming the primary `system.journal` it finds there. `$MACHINE_ID` is an environment variable that you can insert into the Source's search path. If you use the environment variable but don't assign a value, the Source will look up the local host `machine id` from `/etc/machine-id` and use that value.

Currently, this Source does not support all of the compression options that the journald service supports. The parser this Source uses to read the files can handle `LZ4` and `ZSTD` compression. When the parser encounters compressed fields it can't unpack, an error message emits in the logs. Below is an example of the error message:

```
Error reading journald event, dropping offset=31678152 evt="Mar 02 21:29:54
goats-xps15 multipassd: undefined"
```

> Also, this Source does not support the new `COMPACT` binary file format. The logs will contain an error message when the parser encounters this type of file.

Configuring a Journal Files Source

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 to `Yes` to enable the Source.

Input ID: Enter a unique name to identify this File Monitor Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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`.

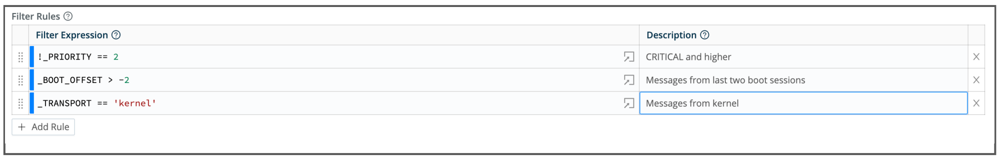
Optional Settings

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 Expression	Description
`!_PRIORITY == 2`	CRITICAL and higher
`_BOOT_OFFSET > -2`	Messages from last two boot sessions
`_TRANSPORT == 'kernel'`	Messages from kernel

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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.

15.10.5. System Metrics

Cribl Stream can collect metrics from the host on which it is running, and can populate some standard metrics dashboards right out of the box. To see all of the metrics this Source supports, check out Linux System Metrics Details.

Type: **System** | TLS Support: **N/A** | Event Breaker Support: **No**

Cribl Edge Workers support System Metrics only when running on Linux, not on Windows.

This Source is not available on Cribl-managed Cribl.Cloud Workers.

Configuring Cribl Stream to Collect System Metrics

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

Configure via QuickConnect

1. In the submenu, click **Collect** (Edge only).
2. Click **Add Source** at left. From the resulting drawer's tiles, select [**System and Internal** >] **System Metrics**.
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 [**System and Internal** >] **System Metrics**.
3. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Optional Settings

Polling interval: How often to collect metrics, in seconds. Defaults to `10`.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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.

Process Metrics

With Process Metrics enabled, Cribl Stream captures process-specific metrics from Linux servers and reports them as events. This allows you to monitor specific processes on Cribl.Cloud instances. You can generate events for any process object.

To collect a process metrics event, create a Process Set and add a filter expression. Processes that match the filter are returned as individual events. See Collecting Metrics. Process Sets are separate from aggregate and host-wide metrics.

Process-specific metrics are **not** affected by the **Host Metrics** detail setting.

Adding a Process Set

To add a Process Set:

1. Open the System Metrics Source and access the **Configure** tab.
2. Click the **Process Metrics** menu, then **Add process set**.
3. Configure the details:
    - **Set name**: The name for this process set.
    - **Filter expression**: The JavaScript expression that will filter the processes.
    - **Include children**: When toggled to “Yes”, the processes that match the filter include metrics for child processes.

Filtering Processes

You can filter processes using the field names and values from each process object.

Tip

To see all processes currently running on an Edge Node, click **Explore** in the submenu and open the Processes Tab.

Example filters

This filter looks for Java processes in the sleep state (an interruptible wait) that are using more than 100% of the CPU or more than 25% of memory:

```
cmdline.args[0] == 'java' && status == 'S' && (cpu > 100 || memory_percent > 25)
```

This filter looks for running NGINX web servers that have more than 12 open file descriptors:

```
service == 'nginx' && fds > 12
```

Here are some more Linux processes you could find metrics for:

- `cmdline.args[0].endsWith('/java') && cmdline.args[1] == 'nifi'`
- `exe.path == '/usr/bin/sshd'`
- `uid == 2`
- `cgroup.0 == /user.slice/user-1000.slice/`
- `stat.status == R`

- `stat.starttime > 943517`
- `cpu > 30`
- `memory_percent > 30`
- `stat.num_threads > 10`

The **Filter expression** field expects a specific syntax in order to apply the filter to the processes. These expressions evaluate to either `true` or `false`. See Filters for more information.

Collecting Metrics

Once you have at least one Process Set created and saved, Cribl Stream will begin collecting process metrics at the interval defined in the **Polling interval** field (**General Settings** menu).

To view the status of the Collector and total events collected, click the **Status** tab in the Linux Metrics Source.

To view a live capture of individual metrics gathered from processes that match the Process Set, click the **Live Data** tab. The Process Set that owns each metric is represented by the `__process_set` internal field.

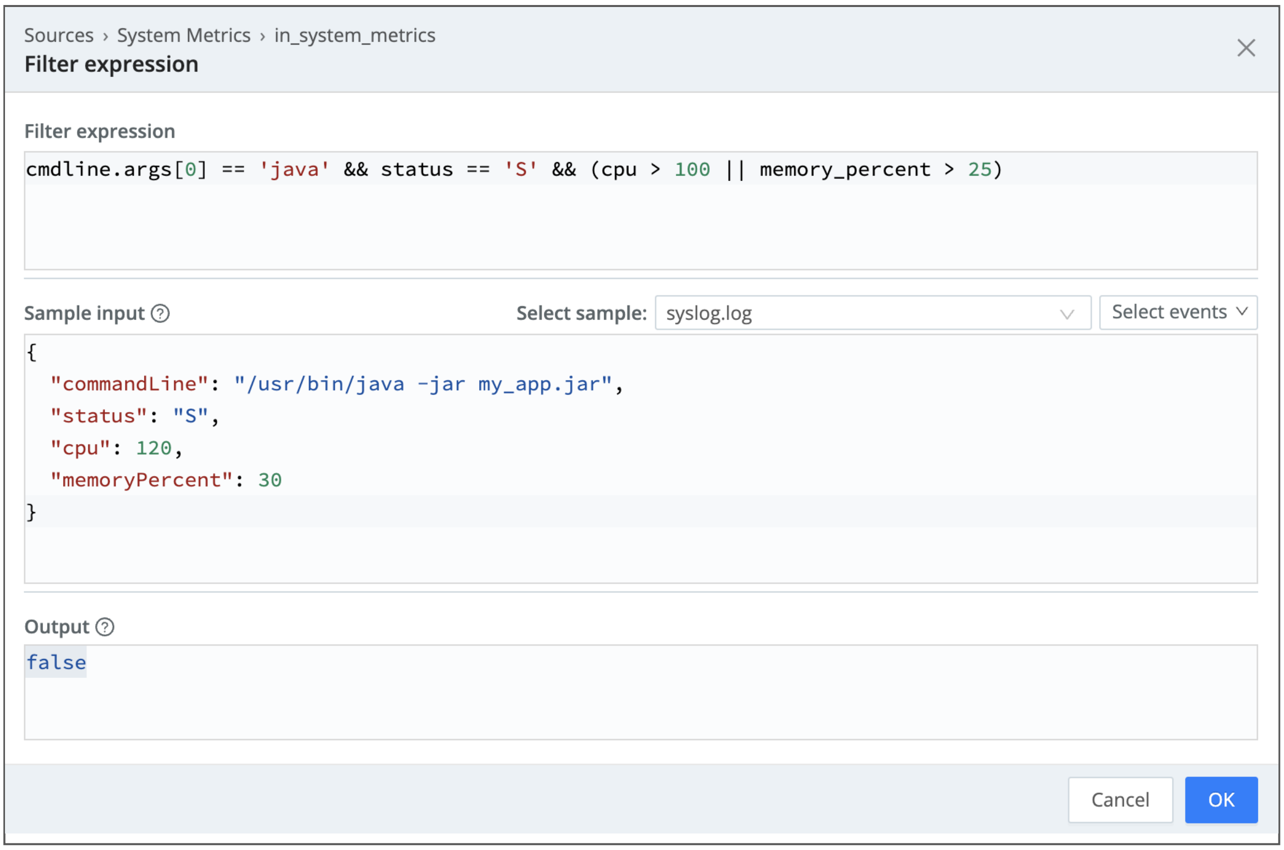
Supported Processes

To view the complete table of supported Linux process metrics, their descriptions, types, and dimensions, see Process-Specific Metrics.

Using Advanced Mode

To test your filter expression against a sample input, click the **Advanced mode** button within the **Filter expression** field. The **Filter expression** modal will open.

Here, you can paste in your filter expression, select a sample JSON input from the drop-down (or enter your own directly in the **Sample input** field), and select a specific event to test against.



```
cmdline.args[0] == 'java' && status == 'S' && (cpu > 100 || memory_percent > 25)
```

```
{
  "commandLine": "/usr/bin/java -jar my_app.jar",
  "status": "S",
  "cpu": 120,
  "memoryPercent": 30
}
```

```
false
```

The **Advanced mode** window, for process-specific metrics

The **Output** will show you if the filter expression returns any matching processes, based on the filter expressions and JSON input.

Container Metrics

Use the buttons to select a level of detail.

- **Basic** generates the server event and per running container events.
- **All** adds per-device and detailed metrics.
- **Custom** provides controls for customizing which containers to generate metrics from.
- **Disabled** means that no metrics will be generated.

The **Custom** buttons displays additional controls, as follows:

- **Container Filters**: Enter a filter expression to govern which containers to generate metrics from. Leave empty (the default) to generate metrics from all containers.
- **All containers**: Toggle to `Yes` to include stopped and paused containers.
- **Per device metrics**: Toggle to `Yes` to generate separate metrics for each device.
- **Detailed metrics**: Toggle to `Yes` to generate full container metrics.

The **Basic**, **All**, and **Custom** buttons provide the following **Advanced Settings**, which are different from those in the main **Advanced Settings** tab:

- **Docker socket**: Enter the full path(s) for Docker's UNIX domain socket. Defaults to `/var/run/docker.sock` and `/run/docker.sock`.
- **Docker timeout**: Timeout, in seconds, for the Docker API. Defaults to `5`.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

If desired, choose a **Pipeline** from the drop-down if you want to process data from this Source before sending it through the Routes.

Disk Spooling

Enable disk persistence: Whether to save metrics to disk. When set to `Yes`, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Stream 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 Stream 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 Stream 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.

15.10.6. 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** | TLS Support: **N/A** | Event Breaker Support: **No**

Cribl Stream supports configuring only one System State Source per Worker Node and per Worker Group.

This Source is currently unavailable on Cribl-managed Cribl.Cloud Workers.

Configuring Cribl Stream 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 Routing > QuickConnect (Stream) or Collect (Edge). 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Collector Settings

Cribl Stream contains the following System State collectors:

- Host Info
- Disks and File Systems
- DNS
- Firewall
- Hosts File
- Interfaces
- Listening Ports
- Logged-In Users
- Routes
- Services
- Users and Groups

Host Info

With the **Enabled** toggle on (the default), Cribl Stream 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** toggle on (the default), Cribl Stream 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 Worker Groups, but you must create a different Worker Group for each OS. Follow this procedure:

1. Create a Worker Group for the appropriate OS.
2. Modify the Worker Group 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 Worker Groups, and remove Linux-specific Sources from Windows Worker Groups.
3. Modify the Worker Group mapping to filter by platform – `platform=='win32'` for Windows or `platform=='linux'` for Linux.
4. Deploy Cribl Stream Worker Nodes and verify that the mapping is correct.

If you are using the free version of Cribl Stream, you will only get a single default Worker Group. If you need more than one Worker Group, you'll have to limit your Cribl Stream deployment to one OS or upgrade to Cribl Stream Enterprise.

For each physical disk, Cribl Stream will collect:

- name
- size (in blocks)
- blockSize (in bytes)
- model
- serial
- firmware
- partitions (count)
- interface type

For each partition on a physical disk, Cribl Stream 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 Stream will collect:

- mountpoint (path)
- disk (name)
- partition (ID)
- filesystem type
- size (in blocks)
- blockSize (in bytes)
- blocksFree
- inodes
- inodesFree
- flags

DNS

With the **Enabled** toggle on (the default), Cribl Stream will create events for DNS resolvers and search entries. The event field mapping depends on the operating system.

Linux

Cribl Stream 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 Stream emits events with the following fields:

Event Field	Description
`ifAlias`	Interface alias name (if present).
`ifIndex`	Interface index (if present).
`addressFamily`	IPv4 or IPv6.
`nameServer`	Array of name servers.
`search`	Array of search suffixes .
`source`	The source of the data used to populate the event.

Hosts File

With the **Enabled** toggle on (the default), Cribl Stream will collect entries from the `hosts` file and emit them as events.

Examples of entries in the `hosts` file are:

- `ip`: IP address of the host.
- `hostnames`: list of hostnames per IP address.

Firewall

With the **Enabled** toggle on (the default), Cribl Stream will collect events from host's defined firewall rules.

Chain Policy/References Rule(s) Event Fields

Cribl Stream 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 Stream emits events with the following fields:

- `chain`: Name.
- `num`: Rule sequence number.
- `pkts`: Number of matched messages processed.
- `bytes`: Cumulative packet size processed (bytes).
- `target`: If the message matches the rule, the specified target is executed.
- `prot`: The specified protocol can be one of `tcp`, `udp`, `udplite`, `icmp`, `icmpv6`, `esp`, `ah`, `sctp`, `mh`, or the special keyword `all`.
- `opt`: Rarely used, this column is used to display IP options.
- `in`: Inbound network interface.
- `out`: Outbound network interface.
- `src`: The source IP address or subnet of the traffic, the latter being anywhere.
- `dest`: The destination IP address or subnet of the traffic, or anywhere.
- `match`: `ctstate`, `state`, `ADDRTYPE` info added to a state field.
- `comment`: Comment defined on the rule.
- `prot_family`: `IPv4` or `IPv6`.

Windows

On Windows, this collector emits events with the following fields:

- `ruleName`: Rule Name.
- `displayName`: Display Name.
- `direction`: Specifies which direction of traffic to match with this rule. The acceptable values for this parameter are `Inbound` or `Outbound`.
- `enabled`: `True` or `False`.
- `action`: Specifies the action to take on traffic that matches this rule. The acceptable values for this parameter are `Allow` or `Block`.
- `group`: Specifies the rule group (if applicable).
- `icmpType`: Specifies the `ICMP` type used.
- `policyStoreSource`: Contains a path to the policy store where the rule originated.
- `profile`: Profile conditions associated with a rule.
- `protocol`: protocols associated with firewall and `IPsec` rules.
- `localPort`: Rules for local-only port mapping.
- `remotePort`: Rules for remote port mapping.

Interfaces

With the **Enabled** toggle on (the default), Cribl Stream will create events for each of the host's network interfaces.

Cribl Stream will create separate events for each entry. All events will have identical timestamp values.

Linux IPV4/IPV6

On Linux, Cribl Stream 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 Stream creates `interface` events with the following fields:

- `ifIndex`: Interface index identification numbers.
- `interface`: The name of the network interface.
- `mtu`: Maximum transmission unit (MTU) in bytes for packets sent on this interface.
- `flags`: Flags.
- `macAddress`: Media Access Control address.
- `addrInfo`: Array of `{family, ipAddress, mask, prefix}`.

Listening Ports

With the **Enabled** toggle on (the default), Cribl Stream will create events from the list of listening ports and their associated process identifier (pid).

Cribl Stream will create separate events for each entry. All events will have identical timestamp values.

Sample entries include:

- `Protocol Family`: `ipv4` or `ipv6`.
- `Protocol`: `TCP`, `UDP`, or `UNIX`.
- `Socket`: `${addr}:${port}`, or `${path}` for UNIX sockets.
- `Process id`: The process identifier.
- `Program`: The program listening on the port.

Logged-In Users

With the **Enabled** toggle on (the default), Cribl Stream will collect entries from currently logged-in users.

Linux

Cribl Stream 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 Stream emits events with the following fields:

- `userName`: Login name of the user.
- `sessionName`: `rdp` (Remote desktop/terminal services login session) or `console` (Direct login session).
- `id`: Session ID.
- `state`: `active` (Active session) or `disc` (Disconnected/inactive session).
- `logonTime`: Date and time the user logged on.
- `source`: Set to `Query User`.

Routes

With the **Enabled** toggle on (the default), Cribl Stream will collect entries from host’s network routes and emit them as events.

Sample entries include:

- `address_family`: `ipv4` or `ipv6`.
- `destination`: The destination network or host.
- `prefix`: Emits only from Linux IPv6.
- `flags`: Flags.
- `gateway`: The gateway address.
- `interface`: Interface to which packets for this route will be sent.
- `ifIndex`: Interface ID. Emits only from Windows.
- `mask`: The netmask for the destination net. Emits only from Linux IPv4.
- `metric`: The distance to the target.
- `sequence`: Sequence number in the table.
- `source`: The source of the data used to populate the event.

Services

With the **Enabled** toggle on (the default), Cribl Stream 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** toggle on (the default), Cribl Stream will create events for users and groups.

Linux

On Linux, Cribl Stream 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 Stream also creates `group` events with the following fields:

- `name`
- `groupId`
- `users`

Windows

On Windows, Cribl Stream creates `user` events with the following fields:

- `userName`
- `fullname`
- `description`
- `groups`

Cribl Stream also creates `group` events with the following fields:

- `name`
- `description`
- `users`

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

Select a **Pipeline** (or Pack) from the drop-down to process this Source's data. Required to configure this Source via **Data Routes**; optional to configure via **Collect**/**QuickConnect**.

Disk Spooling

Enable disk spooling: Whether to save metrics to disk. When set to `Yes`, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Stream 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 Stream 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.

15.11. 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. Note that AppScope is no longer being actively developed by Cribl.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Stream to Receive AppScope Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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.

Downloading AppScope

Cribl Stream must download the AppScope package from the Cribl CDN the first time it performs an operation on this Source. This will also update the AppScope package's version, if a newer one is available in the CDN.

If Cribl Stream cannot access the CDN (for example, if you are working on an airgapped instance), you should manually download the binary and place it in `$CRIBL_HOME/state/download/scope`.

General Settings

Input ID: Enter a unique name to identify this AppScope Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.s

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 Stream will use the runtime user's default permissions.

When **UNIX domain socket** is set to `No`, you instead see the following two fields to specify a network host and port.

- **Address**: Enter the hostname/IP on which to listen for AppScope data. (E.g., `localhost`.) Defaults to `0.0.0.0`, meaning all addresses.
- **Port**: Enter the port number to listen on.

By default:

- In Cribl Stream, **UNIX domain socket** is set to `No`, with default network connections (address and port) of `0.0.0.0:10090` for TCP, and `0.0.0.0:10091` for TLS, respectively.
- In Cribl Edge, **UNIX domain socket** is set to `Yes`, with a UNIX socket path of `$CRIBL_HOME/state/appscope.sock`.

Authentication Settings

Use the **Authentication method** drop-down 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 Stream'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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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** toggle is set to `No`. It provides the following options.

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `Yes`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

AppScope Rules

The AppScope Rules settings are available:

- In Cribl Stream single-instance – but not Cribl.Cloud – deployments.
- In Cribl Edge, both single-instance and Cloud.

Rules: Click **Add Rule** to include processes to scope, and to link to an AppScope config. Once you have saved the configuration, and committed and deployed your changes, AppScope will instrument any process that matches a Rule, on any Edge Node in the Fleet.

(When no Rules are defined, you can still scope by PID in the Edge Processes page. Scope by PID only instruments a single process running in one Edge Node.)

- **Process name:** Matches if the string value you enter corresponds to the basename of the scoped process.
- **Process argument:** Matches if the string value you enter appears as a substring anywhere in the scoped process' full command line (including options and arguments).
- **AppScope config**: Select an AppScope config.

Transport override: Enter a URL to override aspects of the transport configuration, such as the hostname, port, or TLS settings. For details, see Transport Override Details.

Transport Override Details

In scenarios like the following, use the **Transport override** option to extend the defaults in AppScope's transport configuration:

- When this Source is set to **TCP** mode, it typically listens on the default address `0.0.0.0`. Scoped clients will need a specific `IP` or `hostname` to connect. In these cases, set **Transport override** URL to a specific IP/hostname (example format: `tcp://my.host.name`). This Source will parse the URL and look for the `hostname` and `port`, then use those values to override what would otherwise be sent to the `scope start` command.
- When this Source is set to **UNIX domain socket**, it listens by default on `$CRIBL_HOME/state/appscope.sock`. The socket is often created on a mounted volume in a container. The path to that socket might be different outside the container, or when mounted into another container. In these cases, set the **Transport override** URL to specify an alternative path (example format: `unix:///some/other/volume/appscope.sock`).

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.

> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maxiumum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Disk Spooling

For Cribl Search to access the data that arrives at an AppScope Source, **Disk Spooling** must be enabled.

Enable disk persistence: Whether to save metrics to disk. When set to `Yes`, exposes this section's remaining fields.

Bucket time span: The amount of time that data is held in each bucket before it's written to disk. The default is 10 minutes (`10m`).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl Stream 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 Stream 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 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.

Fields for this Source:

- `__inputId`
- `__srcIpPort`

Examples

The following examples work only in Cribl Stream. You can vary the scoped commands (`ps -ef` and `curl`) as desired.

Cribl.Cloud – TLS

An `in_appscope_tls` TLS Source is preconfigured for you on Cribl.Cloud, using port `10090`. You can send it AppScope data using this command:

```
./scope run -c <Your-Ingress-Address>:10090 -- ps -ef
```

Cribl Cloud – TCP

An `in_appscope_tcp` TCP Source is preconfigured for you on Cribl.Cloud, using port `10091`. You can send it AppScope data using this command:

```
./scope run -c tcp://<Your-Ingress-Address>:10091 -- curl -so /dev/null \
https://wttr.in/94105
```

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

This kind of periodic logging helps you determine whether a Source is in fact still healthy even when no data is coming in.

15.12. CrowdStrike FDR

Cribl Stream supports receiving data from the CrowdStrike Falcon Data Replicator (FDR) platform. CrowdStrike data can then be sent to SIEM, threat-hunting, and other security tools and platforms.

This page covers how to configure the Source. Because the FDR platform pulls data from Amazon S3 buckets maintained by CrowdStrike, some of the configuration described here actually involves S3.

Type: **Pull** | TLS Support: **Yes** | Event Breaker Support: **Yes**

(In Cribl Stream 3.2 through 3.5.x, this Source was labeled simply **CrowdStrike**.)

Configuring a CrowdStrike Source

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Pull** >] **CrowdStrike**. 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** >] **CrowdStrike**. 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., `Endpoint42Investigation`.

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'`. The value must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. The expression can only be evaluated at init time, for example when referencing a Global Variable like this: `https://host:port/myQueue-${C.vars.myVar}`.

Optional Settings

Filename filter: Regex matching file names to download and process. Defaults to: `.*`.

Region: AWS Region where the S3 bucket and SQS queue are located. Required, unless **Queue** is a URL or ARN that includes a Region.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the buttons to select an 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 Stream 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 a secret key pair that you've configured in Cribl Stream's internal secrets manager or (if enabled) an external KMS. Follow the **Create** link if you need to configure a key pair.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for S3: Whether to use Assume Role credentials to access S3. Defaults to `Yes`.

Enable for SQS: Whether to use Assume Role credentials when accessing SQS (Amazon Simple Queue Service). Defaults to `No`.

AWS account ID: SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

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`. Toggle to `Yes` to enable the custom command.

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 to the command. You can drag arguments vertically to resequence them.

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. 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 the field's value (can be a constant).

Pre-Processing

Optionally, use the drop-down to select an existing Pipeline to process data from this Source before sending it through the Routes. Otherwise (by default), events will be sent to normal routing and event processing.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

Checkpoint Settings

Enable checkpointing: When toggled to `No` (the default) Cribl Stream will always start processing a file from the beginning. Toggle to `Yes` if, after an error or restart occurs, you want Cribl Stream to resume processing a given file from the point where it was interrupted, as explained below. When turned on, the following additional setting becomes available:

- **Retries**: The number of times you want Cribl Stream to retry processing a file after an error or restart occurs. If **Skip file on error** is enabled, this setting is ignored.

Other Settings

Endpoint: The S3 service endpoint you want CrowdStrike to use. If empty, Cribl Stream 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 Stream will automatically add it to the URL path. For details, see AWS' Path-Style Requests topic.

Signature version: Signature version to use for signing S3 requests. Defaults to `v4`; `v2` is also available.

Num receivers: The number of receiver processes to run. The higher the number, the better the throughput, at the expense of CPU overhead. Defaults to `1`.

Max messages: The maximum number of messages that SQS should return in a poll request. Amazon SQS never returns more messages than this value. (However, fewer messages might be returned.) Acceptable values: `1` to `10`. Defaults to `1`.

Visibility timeout seconds: After messages are retrieved by a `ReceiveMessage` request, Cribl Stream will hide those messages from subsequent retrieve requests for at least this duration. Default is `21600` seconds (6 hours); maximum allowed threshold is `43200` sec. (12 hours). Set to `0` for no message hiding.

Cribl Stream will automatically extend this timeout until the initial request’s files have been processed – notably, for large files that require additional processing time. The generous 6-hour default allows each Worker Process time to decompress, stitch, and read large multi-part files, without duplicate processing.

Poll timeout (secs): How long to wait for events before polling again. Minimum `1` second; default `10`; maximum `20`. Short durations increase the number - and thus the cost – of requests sent to AWS. (The UI will show a warning for intervals shorter than `5` seconds.) Long durations increase the time the Source takes to react to configuration changes and system restarts.

Socket timeout: Socket inactivity timeout (in seconds). Increase this value if retrievals time out during backpressure. Defaults to `300` seconds.

Skip file on error: Toggle to `Yes` to skip files that trigger a processing error. (E.g., corrupted files.) Defaults to `No`, which enables retries after a processing error.

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.

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.

Checkpointing

When Workers restart, checkpointing can prevent the Source from losing data or collecting duplicate events. Among the situations where this works well:

- When Cribl Stream pushes a new configuration: the Leader pushes the configuration to the Workers, and restarts them.
- When a Worker has gotten into a bad state, and you want to restart it.

Checkpointing allows Cribl Stream to resume processing a file at the point where an error or restart occurred, minimizing the number of duplicate events sent downstream in the event of an interruption. To accomplish this, Cribl Stream keeps track of how many events have been processed for a given file. Resuming processing after an interruption, the Source skips the first *N* events it has seen and starts processing from event *N*+1.

The Source manages checkpoint information in memory, but writes it out to disk periodically (once per second, and on Worker shutdown) so that the information can survive a Worker restart. Cribl Stream keeps checkpoint information at the Worker Process level and does not share it between Worker Processes or Worker Nodes.

Checkpointing is a feature to keep in mind when you do disaster recovery (DR) planning.

Caveats to using checkpointing include: Whenever Worker Node state is destroyed (that is, when a Worker ceases to exist), checkpoint information will be lost. In Cribl.Cloud deployments, this can happen during upgrades. Also, because checkpoints are not written to disk on **every** event, the checkpoints written to disk will lag behind those that the S3 Source manages in memory. If the Worker Process stops abruptly without a clean shutdown, checkpoints not yet written to disk will be lost. When processing resumes, the Source will start from just after the last checkpoint written to disk, producing duplicate events corresponding to the lost checkpoints.

Located in the UI at **Advanced Settings** > **Checkpoint Settings** > **Enable checkpointing**, this feature is turned off by default.

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.

How Cribl Stream Pulls Data

Worker Processes poll for messages from SQS, using the AWS SDK S3 APIs. Each polling call will return (by default) one message, if any are available. You can change this default using `Max messages.`

If no message is available, the call will time out after one second, and then polling will repeat.
All Worker Processes participate in polling, so as to disribute files' collection and processing evenly across Workers.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

15.13. 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 Stream 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 Stream 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 Stream. Cribl Stream can parse, filter, and enrich that data, and then send it to any supported Destination, including a Cribl Stream Datadog Destination. (By default, Datadog Agent sends data only to Datadog.)

For inbound log data, this Source supports gzip -compression when the `Content-Encoding: gzip` connection header is set. For other data types, it assumes that all inbound data is compressed using `deflate`.

Configure Datadog to use Datadog API v1

This Source communicates with the Datadog API. Although v2 is the latest version of this API, this Source uses v1, which Datadog still supports.

To configure your Datadog Agent to work with this Source, ensure that in the `datadog.yaml` file, `use_v2_api.series` is set to `false`. Otherwise, when Datadog Agent sends metrics, Cribl Stream will not receive them, and `API Key invalid, dropping transaction` errors will appear in the Datadog Agent logs.

Potential Cribl Stream support for Datadog API v2 is being tracked as **CRIBL-18281** in Known Issues.

Configuring Cribl Stream to Ingest Datadog Agent Output

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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`.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__agent_api_key`
- `__agent_event_type`

- `__final`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`
- `__srcIpPort` – See details below.
- `_time`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Datadog Agent sending data to this Source.

When any proxies (including load balancers) lie between the Datadog Agent and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **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 Stream

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 Stream, you'll set the following environment variables:

- `DD_DD_URL`: The URL or IP address and port of your Datadog Agent Source in Cribl Stream.
- `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 Stream 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 Stream. 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 Stream-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 Stream 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`.	`DD_APM_DD_URL`

Type(s) of Information	Dependencies and/or Limitations	Environment Variable
	Cannot be sent to Cribl Stream even when you **are** sending other types there.	

Managing API Keys

Suppose your data flow runs from Datadog Agents, to Cribl Stream Datadog Agent Sources, to Cribl Stream 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 Stream Datadog Destination.

Another possibility is that you want data to flow to some Destination other than a Cribl Stream 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 Stream 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 Stream 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 Stream Datadog Destination.
- No matter which host the data originates from, your Cribl Stream 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 Stream 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 Stream Datadog Agent Source. Agents within different subgroups of hosts will send different API keys, as described above.
  - Cribl Stream Datadog Agent Source passes the API key from the Agent to the Cribl Stream Datadog Destination as an internal field.
  - Cribl Stream 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 Stream Datadog Agent Source.
- If you have a Cribl Stream Datadog Destination set up, connect your Cribl Stream 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 Stream Datadog Agent Source to connect to some Destination other than a Cribl Stream Datadog Destination. That's beyond the scope of this example, but verifying data flow should be similar.
- If the Cribl Stream 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 Stream.
- 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.

15.14. Elasticsearch API

Cribl Stream 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 Stream to Receive Elasticsearch Bulk API Data over HTTP(S)

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 Stream 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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.

See also Periodic Logging for information on how auth tokens affect product logging.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `Yes`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream 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 Stream'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 Stream 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 Stream 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 Stream 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.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source’s data via the Routing table.

Select **QuickConnect** to send this Source’s data to one or more Destinations via independent, direct connections.

Field Normalization

The Elasticsearch API input normalizes the following fields:

- `@timestamp` becomes `_time` at millisecond resolution.
- `host` is set to `host.name`.
- Original object `host` is stored in `__host`.

The Elasticsearch Destination does the reverse, and it also recognizes the presence of `__host`.

Internal Fields

Cribl 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.

Fields for this Source:

- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__host`
- `__id`
- `__index`
- `__inputId`
- `__pipeline` - If present in the Elasticsearch `event.action` field of the event. See also how to override the pipeline attribute.
- `__srcIpPort` – See details below.
- `__type`

Overriding the Pipeline Attribute in the Elasticsearch and Elastic Cloud Destinations

The Elasticsearch Source will record the pipeline attribute received in a variable named `__pipeline`, if pipeline was presented in the Source event. If you want to forward the pipeline from the Source event to an Elasticsearch or Elastic Cloud Destination, set up the **Elastic pipeline** in one of the following ways.

This expression will use the value of `__pipeline` or default to `'myPipeline'` if `__pipeline` is missing:

```
__pipeline || ‘myPipeline'
```

An alternative is to pass the pipeline if present, or otherwise omit it:

```
__pipeline ? __pipeline : undefined
```

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Elasticsearch client sending data to this Source.

When any proxies (including load balancers) lie between the Elasticsearch client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **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 Stream). The Beats most popular with Cribl users are Filebeat and Winlogbeat.

To set up a Beat to send data to Cribl Stream, 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 Stream Elasticsearch Source endpoint as the Beat's Elasticsearch output. To the Beat, Cribl Stream 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 Stream).

When you are sending from an Elastic Agent to Cribl Stream, the `elastic-agent.yml` file doesn't pay any attention to the settings for `output.elasticsarch.allow_older_versions: true`. As a result, the

Elasticsearch API Source will not get any data.

To set it up correctly, go to the **Advanced Settings** tab, and change the **API Version** to `Custom`. In the **Custom API Version** editor, edit the `version : number` to match the version of the Elastic Agent you are using. This should allow the data to start flowing to the Source.

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

If one or more auth tokens are configured and enabled, Cribl Stream logs requests and events for each enabled auth token individually. Since the tokens themselves are redacted for security, Cribl Stream logs the initial text of the token description to help you identify which token a given log is for.

If no auth token is configured and enabled, Cribl Stream simply logs overall statistics about incoming requests and ingested events.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

15.15. HTTP/S (Bulk API)

Cribl Stream 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 Stream to Receive Data over HTTP(S)

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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**.

See also Periodic Logging for information on how auth tokens affect product logging.

Optional Settings

Cribl HTTP event API: Base path on which to listen for Cribl HTTP API requests. To construct the actual endpoint, Cribl Stream 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 Stream 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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 Stream 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":"r
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"me
```

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 Stream

The examples in this section demonstrate sending HTTP data into a Cribl Stream 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 Stream receiver.

Cribl Endpoint – Single Event

Cribl Single Event Example:

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d '{'
```

Cribl Endpoint – Multiple Events

Cribl Endpoint - Multiple Events

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d $'
```

Splunk HEC Event Endpoint

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization: myT

curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42
```

For Splunk HEC, the token specification can be either `Splunk <token>` or `<token>`.

Cribl.Cloud – Single Event

1. Generate and copy a token in your Cribl.Cloud instance's HTTP Source > **General Settings**.
2. From the command line, use `https`, your Cribl.Cloud portal's **Ingest Endpoint** and port, and the token's value:

Cribl.Cloud – Single Event

```
curl -k https://default.main-<Your-Org-ID>.cribl.cloud:10080/cribl/_bulk -H 'Autho
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Worker Groups.

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

If one or more auth tokens are configured and enabled, Cribl Stream logs requests and events for each enabled auth token individually. Since the tokens themselves are redacted for security, Cribl Stream logs the initial text of the token description to help you identify which token a given log is for.

If no auth token is configured and enabled, Cribl Stream simply logs overall statistics about incoming requests and ingested events.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source’s **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

15.16. Raw HTTP/S

Cribl Stream 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 Stream to Receive Raw HTTP Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

> You can optimize Workers' startup connections and CPU load at **Group 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.

Capture request headers: Toggle this to `Yes` to add request headers to events, in the `__headers` 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.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Max requests per socket: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing Connection Reuse Against Request Distribution below.

Socket timeout (seconds): How long Cribl Stream 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 Stream 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 Stream will reuse connections. The shorter the timeout, the closer Cribl Stream gets to creating a new connection for every request. When request frequency is high, you can use longer timeouts to reduce the number of connections created, which mitigates the associated cost.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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.)

15.17. Metrics

Cribl Stream 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 Stream to Receive Metrics

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

IP allowlist regex: Regex matching IP addresses that are allowed to send data. Defaults to `.*` (i.e., all IPs.)

**Max buffer size (events) **: Maximum number of events to buffer when downstream is blocking. Defaults to `1000`.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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 Stream 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.

15.18. Model Driven Telemetry

Cribl Stream supports receiving network device metrics and events via Model Driven Telemetry (MDT).

Type: **Push** | TLS Support: **Yes** | Event Breaker Support: **No**

For observability of network devices, MDT compares favorably with the older SNMP-based approach because:

- MDT's messaging granularity is much finer than SNMP's.
- MDT's push-based model reduces load on network devices, while SNMP's pull-based model queues requests on devices.
- MDT's extensive ecosystem of data modules enables devices to collect many more kinds of data than they could with SNMP tools.

Prerequisites

When you configure MDT on a network device you choose among a variety of options and techniques. For this Source (in Cribl Stream version 4.5), you must configure your upstream sending device to use the following:

- MDT in dial-out mode.
- gRPC for the data transport mechanism.
- YANG models for data representation.
- Self-describing, also known as Key/Value Protocol Buffers (protobufs) for data encoding.

Configuring Cribl Stream to Receive Model Driven Telemetry Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Next, click **Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **Model Driven Telemetry**. 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** >] **Model Driven Telemetry**. 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: Enter the port number to listen on. Defaults to port `57000`.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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`.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (such as the system's CA). Defaults to `Yes`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If

the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 of disk space allocated per PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of `KB`, `MB`, and so forth. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, and so forth. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`. This setting helps ensure events the best chance of delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Worker startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Max active connections: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Shutdown timeout: Time to allow the server to shut down gracefully before forcing shutdown. Defaults to `5000` milliseconds (5 seconds).

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of data from this Source via the Routing table.

Select **QuickConnect** to send data from this Source to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__srcIpPort`

15.19. NetFlow

Cribl Stream supports receiving NetFlow v5 data via UDP.

Type: **Push** | TLS Support: **No** | Event Breaker Support: **No**

This Source ingests NetFlow records similarly to how it ingests events from other upstream senders: fields are broken out, and the message header is included with each record. If you prefer to render NetFlow data as metrics, use a pre-processing Pipeline or a Route.

Configuring Cribl Stream to Receive NetFlow Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **+ Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **NetFlow**. 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** >] **NetFlow**. Next, click **Add Source** to open a **New Source** modal that provides the options below.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP sender. Cribl also recommends tuning the OS UDP buffer size.

General Settings

Input ID: Enter a unique name to identify this NetFlow Source definition.

Address: Enter the hostname/IP to listen for NetFlow data. For example: `localhost`, `0.0.0.0`, or `::`.

Port: Enter the port number.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.
>
> For more about PQ modes, see Always On versus Smart Mode.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (that is, all IPs).

IP denylist regex: Regex matching IP addresses whose messages you want this Source to ignore. Defaults to `^$` (that is, every specific IP address in the list). This takes precedence over the allowlist.

UDP socket buffer size (bytes): Optionally, set the `SO_RCVBUF` socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Minimum: `256`. Maximum: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

What Fields to Expect

The NetFlow Source does minimal processing of the incoming UDP messages to remain consistent with the internal Cribl event model. For each UDP message received on the socket, you can expect an event with the following fields. We have organized these fields into categories to make them easier to grasp; the categories are our own, and not from the NetFlow specifications. The field definitions are mostly copied from Cisco NetFlow documentation.

General

- `_time`: The UNIX timestamp (in seconds) at which the message was received by Cribl Stream.
- `source`: A string in the form `udp|<remote IP address>|<remote port>`, indicating the remote sender.
- `host`: The hostname of the machine running Cribl Stream that ingested this event.
- `inputInt`: SNMP index of input interface; always set to zero.
- `outputInt`: SNMP index of output interface.
- `protocol`: IP protocol type (for example, TCP = 6; UDP = 17) of the observed network flow.
- `tcpFlags`: Cumulative logical OR of TCP flags in the observed network flow.
- `tos`: IP type of service; switch sets it to the ToS of the first packet of the flow.
- `icmpType`: Type of the ICMP message. Only present if the protocol is ICMP.
- `icmpCode`: Code of the ICMP message. Only present if the protocol is ICMP.

Because this Source reads input data directly from bytes in a compact format, there is nothing suitable to put in a `_raw` field, and the Source does not add a `_raw` field to events.

Flow Source and Destination

- `srcAddr`: Source IP address; in case of destination-only flows, set to zero.
- `dstAddr`: Destination IP address.

- `nextHop`: IP address of next hop router.
- `srcPort`: TCP/UDP source port number.
- `dstPort`: TCP/UDP destination port number. If this is an ICMP flow, this field is a combination of ICMP type and ICMP code, which are broken out separately as `icmpType` and `icmpCode` fields.
- `srcMask`: Source address prefix mask bits.
- `dstMask`: Destination address prefix mask bits.
- `srcAs`: Autonomous system number of the source, either origin or peer.
- `dstAs`: Autonomous system number of the destination, either origin or peer.

Flow Statistics

- `packets`: Packets in the flow.
- `octets`: Total number of Layer 3 bytes in the packets of the flow.
- `startTime`: System uptime, in milliseconds, at the start of the flow.
- `endTime`: System uptime, in milliseconds, at the time the last packet of the flow was received.
- `durationMs`: `endTime` minus `startTime`.

Header

The following are subfields within the `header` field:

- `count`: Number of flows exported in this flow frame (protocol data unit, or PDU).
- `sysUptimeMs`: Current time in milliseconds since the export device booted
- `unixSecs`: Current seconds since 0000 UTC 1970.
- `unixNsecs`: Residual nanoseconds since 0000 UTC 1970.
- `flowSequence`: Sequence counter of total flows seen.
- `engineType`: Type of flow switching engine.
- `engineId`: ID number of the flow switching engine.
- `samplingMode`: The first two bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling mode.
- `samplingInterval`: The remaining 14 bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling interval.
- `version`: NetFlow export format version number.

Also, the internal fields listed below will be present.

Internal Fields

Cribl 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.

Fields accessible for this Source:

- `__bytes`
- `__inputId`
- `_time`

UDP Tuning

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn’t an issue. As you scale up, however, the default size of that buffer may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

A typical value of about 200 KB is far too small for an even moderately busy syslog server. You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means you need to increase your `net.core.rmem_max` setting (see earlier).

You can update the live settings, but you’ll also need to change the boot-time setting so next time you reboot everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

We recommend you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the UDP (Raw) Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also indicate your Worker is encountering pressure further down the Pipeline.
- Especially if you increase your kernel receive buffer as above, watch your Worker processes' memory usage.

15.20. OpenTelemetry (OTel)

Cribl Stream 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 Stream 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 Stream 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 Stream'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 **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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).

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 Stream 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 Stream OTel Source will receive OTel data from a **Collector** running on a local **agent**. In Cribl Stream's terminology, the Collector is the **client** and the OTel Source is the **server**. This is why this Source's UI identifies the Source's TLS Settings as "server-side."

For more about this client-server relationship, see the TLS Configuration Example below.

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs (mTLS). Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (for example, the system's CA). Defaults to `Yes`.
- **Common Name**: Regex that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below,

> use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event, using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Advanced Settings always displays **Extract spans**, **Extract metrics**, and **Environment**:

The **Extract spans** and **Extract metrics** settings are unique to OTel. Their default `No` settings allow Cribl Stream 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.

- **Extract spans**: Toggle to `Yes` if you want Cribl Stream to generate an individual event for each span. (Recall that traces contain multiple spans.)
- **Extract metrics**: Toggle to `Yes` if you want Cribl Stream to generate an individual event for each data point. (Recall that OTel metric events contain multiple data points.)
- **Environment**: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

When **General Settings** > **Protocol** is set to `gRPC`, the UI displays one additional setting:

- **Max active connections**: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

When **General Settings** > **Protocol** is set to `HTTP`, the UI displays five additional settings:

- **Max active requests**: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.
- **Max requests per socket**: The maximum number of requests Cribl Stream should allow on one socket before instructing the client to close the connection. Defaults to `0` (unlimited). See Balancing

Connection Reuse Against Request Distribution below.

- **Socket timeout (seconds)**: How long Cribl Stream 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 Stream 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).

Balancing Connection Reuse Against Request Distribution

Max requests per socket allows you to limit the number of HTTP requests an upstream client can send on one network connection. Once the limit is reached, Cribl Stream uses HTTP headers to inform the client that it must establish a new connection to send any more requests. (Specifically, Cribl Stream sets the HTTP `Connection` header to `close`.) After that, if the client disregards what Cribl Stream has asked it to do and tries to send another HTTP request over the existing connection, Cribl Stream will respond with an HTTP status code of `503 Service Unavailable`.

Use this setting to strike a balance between connection reuse by the client, and distribution of requests among one or more Worker Node processes by Cribl Stream:

- When a client sends a sequence of requests on the same connection, that is called connection reuse. Because connection reuse benefits client performance by avoiding the overhead of creating new connections, clients have an incentive to **maximize** connection reuse.
- Meanwhile, a single process on that Worker Node will handle all the requests of a single network connection, for the lifetime of the connection. When receiving a large overall set of data, Cribl Stream performs better when the workload is distributed across multiple Worker Node processes. In that situation, it makes sense to **limit** connection reuse.

There is no one-size-fits-all solution, because of variation in the size of the payload a client sends with a request and in the number of requests a client wants to send in one sequence. Start by estimating how long connections will stay open. To do this, multiply the typical time that requests take to process (based on payload size) times the number of requests the client typically wants to send.

If the result is 60 seconds or longer, set **Max requests per socket** to force the client to create a new connection sooner. This way, more data can be spread over more Worker Node processes within a given unit of time.

For example: Suppose a client tries to send thousands of requests over a very few connections that stay open for hours on end. By setting a relatively low **Max requests per socket**, you can ensure that the same work is done over more, shorter-lived connections distributed between more Worker Node processes, yielding better performance from Cribl Stream.

A final point to consider is that one Cribl Stream Source can receive requests from more than one client, making it more complicated to determine an optimal value for **Max requests per socket**.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

TLS Configuration Example

Here's a simple example for using TLS to secure the communication between an OpenTelemetry client and your Cribl Stream 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.p
   ```

   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 Stream Worker Node on which your OTel Source is running, or (b) the IP address of the load balancer for the relevant Worker Group. In the example snippet below, this is the `<Cribl_IP_address>`. Specify the port on which Cribl Stream'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 Stream 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 Stream 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
```

15.21. SNMP Trap

Cribl Stream supports receiving data from SNMP Traps.

Type: **Push** | TLS Support: **NO** | Event Breaker Support: **No**

Configuring Cribl Stream to Receive SNMP Traps

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream, except in Cribl.Cloud, ships with an SNMP Trap Source preconfigured to listen on port `9162`. You can clone or directly modify this Source to further configure it, and then enable it.

General Settings

Input ID: Enter a unique name to identify this Source definition. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

Address: Address to bind on. Defaults to `0.0.0.0` (all addresses).

UDP Port: Port on which to receive SNMP traps. Defaults to `162`.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication Settings

SNMPv3 authentication provides secure access to devices through optional user authentication and decryption of incoming data packets. Cribl Stream lets you choose any of the following levels of security:

- User name checking without authentication.
- User authentication without privacy.
- User authentication with privacy.

SNMPv3 authentication: Toggle to `Yes` to enable SNMPv3 authentication and reveal authentication parameters. Defaults to `No`.

Allow unmatched traps: Toggle to `Yes` to pass through traps that don't match any of the configured users. Cribl Stream will not attempt to authenticate or decrypt these traps. When toggled to `No` (default), Cribl Stream drops traps without a correctly configured user name.

v3 Name: Enter the SNMPv3 user name (required). Multiple users are supported.

You must configure at least one user to enable SNMPv3 authentication. If desired, you can enter separate user credentials for each SNMPv3 user. The authentication protocol, authentication key, privacy protocol, and privacy key can be unique for each user.

For authentication to succeed, the user name and authentication protocol and key must match in the Source and the sending device configuration.

For decryption to succeed, the user name, authentication protocol and key, and the privacy protocol and key must match in the Source and the sending device configuration.

Authentication protocol: Select the authentication protocol required for your use case. If you select `None`, Cribl Stream will only check the user name without authenticating the user or decrypting the data. Otherwise, Cribl Stream uses the authentication protocol you specify and the key you provide to authenticate the user.

Options include:

- `None`
- `MD5`
- `SHA`

- `SHA224`
- `SHA256`
- `SHA384`
- `SHA512`

v3 authentication key: Enter the authentication key.

Choosing an authentication protocol also allows you to select an optional privacy protocol to decrypt incoming packets.

Cribl Stream logs authentication failures at the `debug` level.

Privacy protocol: Select the privacy protocol required for your use case. If you select `None`, Cribl Stream authenticates the user without decrypting the incoming data. Otherwise, Cribl Stream uses the privacy protocol you specify and the key you provide to decrypt the data.

Options include:

- `None`
- `DES`
- `AES`
- `AES256b (Blumenthal)`
- `AES2556r (Reeder)`

For every successfully decrypted trap, Cribl Stream adds `__didDecrypt: true` to the Event. Cribl Stream logs decryption failures at the `debug` level.

v3 privacy key: Enter the privacy key.

Add user: Click to add a new set of user credentials for SNMPv3 authentication.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group 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`.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

⚠ Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

Fields for this Source:

- `__didDecrypt`: Set to `true` for every trap that is successfully decrypted.
- `__final`
- `__inputId`
- `__snmpRaw`: Buffer containing Raw SNMP packet
- `__snmpVersion`: Acceptable values are `0`, `2` , or `3`. These respectively indicate SNMP v1, v2c, and v3.
- `__srcIpPort` : In this particular Source, this field uses a pipe (|) symbol to separate the source IP address and the port, in this format: `event.__srcIpPort = ${rInfo.address}|${rInfo.port};`
- `_time`

Considerations for Working with SNMP Trap Data

- It’s possible to work with SNMP metadata (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.

15.22. Syslog

Cribl Stream 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 Stream, see Syslog Best Practices.

Configuring Cribl Stream to Receive Data over Syslog

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP senders. Cribl also recommends tuning the OS UDP buffer size.

Optional Settings

Fields to keep: List of fields from source data to retain and pass through. Supports wildcards. Defaults to `*` wildcard, meaning keep all fields. Fields **not** specified here (by wildcard or specific name) will be removed from the event.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries,

the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data.

Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

> In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.
>
> You can optimize Workers' startup connections and CPU load at **Group 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 Stream 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.

Enable TCP load balancing: Toggle to `Yes` if you want the Source to load balance traffic across all Worker Processes, as explained below. (This setting is available only in Cribl Stream deployed in Distributed mode.)

IP 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.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

Setting this value will affect OS memory utilization.

Default timezone: Timezone to assign to timestamps that omit timezone info. Accept the default `Local` value, or use the drop-down list to select a specific timezone by city name or GMT/UTC offset.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

TCP Load Balancing

This feature is available only for Cribl Stream deployments that are in Distributed mode.

When the Syslog Source receives a high volume of data over TCP, a single Worker Process can place excessive CPU load on its Cribl Stream Worker Node, while remaining Worker Processes on the same node mostly sit idle. This undesirable condition is called "TCP pinning" because the high volume of traffic "pins" a single TCP connection.

To alleviate TCP pinning, try toggling **Advanced Settings** > **Enable TCP load balancing** on. The Syslog Source will then load balance traffic across all Worker Processes.

When TCP load balancing is enabled, the Worker Node forks a special Worker Process dedicated to load balancing; creates sockets for communication between the load balancer and the regular Worker Processes; and, makes TCP load balancing metrics available.

The Load Balancing Process

The Worker Node forks a new, special **load balancer** Worker Process that distributes incoming syslog data among the other Worker Processes. For customer-managed and hybrid Worker Groups, consider reducing the Process count by 1, if possible. This will leave a core free to run the load balancer process.

If you want to verify that the load balancer Worker Process exists, teleport into the relevant Worker and in the **Worker Processes** tab you will see a Worker Process with `LB` in its name.

To view logs from the Leader UI, navigate to **Monitoring** > **Logs** > `<Worker Node ID>` > **Load Balancer**.

Worker Process Socket Files

The load balancer Worker Process sends data to the regular Worker Processes over inter-process communication (IPC) sockets. This means that in the `state` directory, there is a socket file for each load balancer Worker Process and each Worker Process.

By default, the socket files are linked to `state`, from the Cribl `tmp` directory. For customer-managed and hybrid Worker Groups, you can specify a directory other than `tmp` for these links. Cribl recommends that you do **not** use this option, but if you find it necessary, navigate to **Group Settings** > **General Settings** > **Sockets** to configure as desired.

TCP Load Balancing Metrics

Metrics specific to TCP load balancing become available.

- `lb.bytes_out` – Total bytes processed by the load balancer Worker Process, by Source.
- `lb.writable_sockets` – Total unblocked Worker Process sockets, by Source.
- `lb.blocked_sockets` – Total blocked Worker Process sockets, by Source.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

TAG Section Processing

Cribl Stream 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 Stream will also process hyphens that appear in an RFC 3164–formatted Syslog message's `TAG` section. If Cribl Stream 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 Stream will treat that character as the termination of the `TAG` section, and as the starting character of the `CONTENT` section.

`CONTENT` Section Processing

Cribl Stream 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 Stream.
- `__srcIpPort <udp|tcp>:<port>`: See Internal Fields.
- `__syslogFail`: See Internal Fields.
- `host`: IP address of the device sending the incoming message.

If messages sent to the Source comply with either RFC 3164 or RFC 5424, fields that the RFC deems guaranteed will always be there, but fields deemed optional might or might not be. Once Cribl Stream 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 Stream 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 Stream 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 Stream uses a set of internal fields to assist in handling of data. These "meta" fields are **not** part of an event, but are accessible and Functions can use them to make processing decisions.

Fields for this Source:

- `__inputId`
- `__srcIpPort <udp|tcp>:<port>`: Identifies the port of the sending socket.
- `__syslogFail`: `true` for data that fails RFC 3164/5424 validation as syslog format.

`stats` Log Message

In Cribl Stream 4.2.2 and later, `stats` log messages report the number of events received, buffered, or dropped for exceeding the maximum Cribl buffer size. By default, these messages are logged every 60 seconds. These values also appear in **Sources** > **Syslog** > <Source_name> > **Status**, in the **UDP** row.

UDP Tuning

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn't an issue. As you scale up, however, the default size of that buffer may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

A typical value of about 200 KB is far too small for an even moderately busy syslog server. You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means you need to increase your `net.core.rmem_max` setting (see earlier).

You can update the live settings, but you'll also need to change the boot-time setting so next time you reboot everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

We recommend you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the Syslog Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also indicate

your Worker is encountering pressure further down the Pipeline.

- Especially if you increase your kernel receive buffer as above, watch your Worker processes' memory usage.

Troubleshooting 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.

15.23. TCP JSON

Cribl Stream can receive newline-delimited JSON data over TCP.

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Configuring Cribl Stream to Receive TCP JSON Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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** drop-down 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

> On Cribl-managed 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group 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 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`

Format

Cribl Stream 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":'
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"message\":\"Something
```

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 Stream 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 Stream host: `cat test.json | nc <myCriblHost> 10001`

Cribl Stream to Cribl.Cloud

This second example demonstrates using TCP JSON to send data from one Cribl Stream 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 Stream instance's Destination, set the following field values to match the target Cloud instance's defaults:

General Settings

Address: `default.main-<Your-Org-ID>.cribl.cloud` – you can simply copy/paste your Cribl.Cloud portal's **Ingest Endpoint** here. With a Cribl.Cloud Enterprise plan, generalize the `default.main` substring in this URL to `<group-name>.main` when sending to other Worker Groups.

Port: `10070`

TLS Settings (Client Side)

Enabled: `Yes`

Validate server certs: `Yes`

Periodic Logging

Cribl Stream logs metrics about incoming requests and ingested events once per minute.

These logs are stored in the `metrics.log` file. To view them in the UI, open the Source's **Logs** tab and choose **Worker Process X Metrics** from the drop-down, where **X** is the desired Worker process.

This kind of periodic logging helps you determine whether a Source is in fact still healthy even when no data is coming in.

15.24. TCP (Raw)

Cribl Stream supports receiving of data over TCP. (See examples and header options below.)

Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Stream to Receive TCP Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

On Cribl-managed 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings. **Queue file path**: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

> You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Custom Command

In this section, you can pass the data from this input to an external command for processing before the data continues downstream.

Enabled: Defaults to `No`. When toggled to `Yes`:

Command: Enter the command that will consume the data (via `stdin`) and will process its output (via `stdout`).

Arguments: Click **Add Argument** to add each argument for the command. You can drag arguments vertically to resequence them.

Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maxiumum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. When this setting is enabled, the `__srcIpPort` internal field will show the original source IP address and port. When it is disabled, the `__srcIpPort` field will show the IP address and port of the proxy that forwarded the connection.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e,. all IPs).

Max active connections: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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.

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 Stream Cloud sample, copy the `<token_value>` out of your Cribl Stream 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 Stream

1. Configure Cribl Stream 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 Stream host, using this command: `cat test.raw | nc <myCriblHost> 7777`

Enabling the Example – Cribl.Cloud

Use netcat with `--ssl` and `--ssl-verify`:

Command-line test

```
cat test.raw | nc --ssl --ssl-verify default.main-<Your-Org-ID>.cribl.cloud 10060
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Worker Groups.

15.25. UDP (Raw)

Cribl Stream supports receiving raw, unparsed data via UDP.

Type: **Push** | TLS Support: **NO** | Event Breaker Support: **NO**

Configuring Cribl Stream to Receive Raw UDP Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP sender. Cribl also recommends tuning the OS UDP buffer size.

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. For example: `localhost`, `0.0.0.0`, or `::`.

Port: Enter the port number.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Single msg per UDP: Toggle to `Yes` if each UDP message should be treated as an independent event. Leave set to the default `No` if the message should be broken on newlines to create multiple events.

Ingest raw bytes: Toggle to `Yes` to add a `__rawBytes` field to each event containing an array of the bytes received as the UDP message.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e, all IPs).

Max buffer size (events): Maximum number of events to buffer when downstream is blocking. The buffer is only in memory.

UDP socket buffer size (bytes): Optionally, set the SO_RCVBUF socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Min: `256`. Max: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

⚠ Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

What Fields to Expect

The Raw UDP Source does minimal processing of the incoming UDP messages to remain consistent with the internal Cribl event model. For each UDP message received on the socket, you can expect an event with the following fields:

- `_raw`: Contains the UTF-8 representation of the entire message received (if **Single msg per UDP** is set to `Yes`), or of the given line that was split out of the message.
- `_time`: The UNIX timestamp (in seconds) at which the message was received by Cribl Stream.
- `source`: A string in the form `udp|<remote IP address>|<remote port>`, indicating the remote sender.
- `host`: The hostname of the machine running Cribl Stream that ingested this event.

Also, the internal fields listed below will be present.

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.

Fields accessible for this Source:

- `__inputId`
- `__srcIpPort`
- `__rawBytes`: When **Ingest raw bytes** is set to `Yes`, this field will be an array containing the bytes of the UDP message.

UDP Tuning

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn't an issue. As you scale up, however, the default size of that buffer may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

A typical value of about 200 KB is far too small for an even moderately busy syslog server. You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means you need to increase your `net.core.rmem_max` setting (see earlier).

You can update the live settings, but you'll also need to change the boot-time setting so next time you reboot everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

We recommend you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the UDP (Raw) Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also

indicate your Worker is encountering pressure further down the Pipeline.

- Especially if you increase your kernel receive buffer as above, watch your Worker processes’ memory usage.

15.26. Windows Event Forwarder

Cribl Stream 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 Stream to Receive Windows Events

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Source, Cribl Stream will add `-CLONE` to the original **Input ID**.

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 Stream 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 that a peer certificate's subject attribute must match in order to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. (For example, to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.) If the subject attribute contains Subject Alternative Name (SAN) entries, the Source will check the regex against all of those but ignore the Common Name (CN) entry (if any). If the certificate has no SAN extension, the Source will check the regex against the single name in the CN.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Verify certificate via OCSP: If toggled to `Yes`, Cribl Stream 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 Stream will fail checks on **any** OCSP error. Otherwise, Cribl Stream 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 Stream 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 to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 Stream 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 Stream.

> 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 Stream 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 Stream 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 Stream, 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.

Fields: Use this setting to add fields exclusively to the events that WEF delivers **for the subscription you are defining**, using Eval-like functionality. (To add fields to all incoming events regardless of subscription, use **Processing Settings** > **Fields** instead.)

Optionally, you can create the field such that its **Value** is the value of a subscription metadata element. (See the available subscription metadata elements listed below.)

To do this, specify a **Value** of `__subscription.<metadata_element>`. The Source will then find the metadata element you specified in the special object `__subscription` and set the value to that.

For example, if you wanted all events for the subscription to be tagged with its **Locale**, you could set the **Name** to `Locale` (or any arbitrary name) and **Value** to `__subscription.locale`.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Subscription Metadata Elements

Metadata Element	Type	Corresponding UI Setting
`version`	string	**Version**
`subscriptionName`	string	**Name**
`contentFormat`	`Raw` or `RenderedText`	**Format**
`readExistingEvents`	boolean	**Read existing events**
`heartbeatInterval`	number	**Heartbeat**
`batchTimeout`	number	**Batch timeout**
`sendBookmarks`	boolean	**Use bookmarks**
`compress`	boolean	**Compression**
`queries`	array	see About the Query builder mode below

About Query builder mode

This setting produces the `queries` array in one of two different forms, depending on whether you select its **Simple** button or its **Raw XML** button.

Simple produces an array whose elements each contain `path`, a string that corresponds to **Queries** > **Path**, and `queryExpression`, a string that corresponds to **Queries** > **Query expression**.

Raw XML produces an array whose elements each contain only the `xmlQuery` metadata element. This is the XML query you specified, in the form a string, that corresponds to the **XML query** setting.

To access any element in the array, use array indexing. For example:

- `__subscription.queries[0].path` will access the path to the first query in an array created by **Query builder mode** > **Simple**.
- `__subscription.queries[0].xmlQuery` will access the first query in an array created by **Query builder mode** > **Raw XML**.

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 Stream 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 TimeCrea
  </Query>
</QueryList>
```

To use this subscription in a Cribl Stream 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) &lt;= 86400000]]]`

You do not need to use the `<Query>` element's `Id` or `Path` attributes anywhere in the Cribl Stream 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 PQ-enabled Source, per Worker Process.

The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.

- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

In Cribl Stream 4.1 and later, Source-side PQ's default **Mode** is `Always on`, to best ensure events' delivery. For details on optimizing this selection, see Always On versus Smart Mode.

You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

Use this setting to add fields to **all events coming into the Source**, using Eval-like functionality. (To add fields on a per-subscription basis, use **Subscriptions** > **Fields** instead.)

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Allow MachineID mismatch: Set this to `Yes` if you do not want to verify that the events sent by a client match the client's certificate Common Name (Subject CN). If set to `No`, events where the `MachineID` (by default the client's machine name, like `CLIENT1.domainName.com`) do not match the CN will be rejected.

This setting applies only to client certificate authentication. If you use Kerberos authentication, this option will not appear and will be treated as a `No` value.

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 Stream should wait before assuming that an inactive socket has timed out. The default `0` value means wait forever. See also Adjusting Timeouts.

Keep-alive timeout (seconds): After the last response is sent, Cribl Stream how long to wait for additional data before closing the socket connection. Defaults to `90` seconds; minimum is `1` second; maximum is `600` seconds (10 minutes).

The longer the timeout, the more Cribl Stream will reuse connections. The shorter the timeout, the closer Cribl Stream 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.

See also Adjusting Timeouts.

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 authentication so it can avoid the overhead of performing the two-step Kerberos authentication on each new batch delivery.

We recommend that you set **Subscriptions** > **Batch timeout**, **Subscriptions** > **Heartbeat**, or both to be less than or equal to **Keep-alive timeout**, especially when a load balancer is in front of the Worker Nodes.

If the socket timeout has elapsed between event deliveries or heartbeats, Cribl Stream will still handle it gracefully, but incur an additional overhead of two network round trips.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

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.

Fields for this Source:

- `__final`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`
- `__srcIpPort` – See details below.
- `__subscriptionName`
- `__subscriptionVersion`
- `_raw`
- `_time`

Configuring Upstream Clients/Senders

This section summarizes how to configure Windows endpoints/senders to forward events to this Cribl Stream 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 Stream'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 Stream.
   - 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 Stream now according to the configured subscriptions. If they are not:

- Certificate authentication only: Verify that the clients can reach the Cribl Stream instance through the network to port 5986 (or other configured port) via TLS/HTTP. If clients are connecting to Cribl Stream via a proxy, you may need to set **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 Stream 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 Worker Node to which you have shell access. This guide assumes a recent Ubuntu distribution.
- The fully qualified domain name (FQDN) of this example Worker Node is `worker1.contoso.com`.
- You are using a Kerberos credential cache of the `DIR` or `FILE` type.

Initial Setup for the Worker Node

To set up the Worker Node:

1. Ensure that the Worker Node can resolve the domain controller’s FQDN. You can test it with: `ping DC01.contoso.com`.

2. Ensure that the Worker 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 Worker 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 Worker 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 Worker 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 Stream Worker Node. Carefully note the file's location on the Worker Node, because you will need it later. The keytab command and output will resemble this example:

```
PS C:\Users\Administrator> ktpass /princ http/wefkerb.weftest.local@WEFTEST.LOCAL /
Targeting domain controller: SUP-DC01.weftest.local
Using legacy password setting method

Successfully mapped http/wefkerb.weftest.local@WEFTEST.LOCAL to cribl.
Key created.

Output keytab to wefkerb.weftest.local.keytab:
Keytab version: 0x502
keysize 110 http/wefkerb.weftest.local@WEFTEST.LOCAL ptype 1 (KRB5_NT_PRINCIPAL) vn
```

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 Worker 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.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the WEF client sending data to this Source.

In the WEF Source configuration, there's an option to handle the `X-Forwarded-For` header, which is commonly used by load balancers to pass the original client's IP address. When any proxies (including load balancers) lie between the WEF client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **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.)

Adjusting Timeouts

For best results, adjust the following timeouts in **Advanced Settings**:

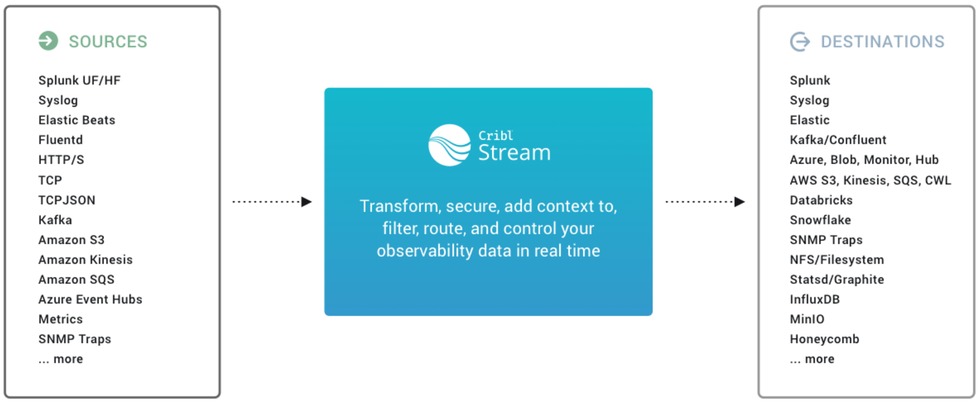
- **Socket timeout** must be higher than **Subscriptions** > **Batch timeout** (or the lowest **Batch timeout** if several subscriptions are used).
- **Keep-alive timeout** must be higher than **Subscriptions** > **Batch timeout** (or the lowest **Batch timeout** if several subscriptions are used).
- **Keep-alive timeout** must be higher than **Subscriptions** > **Heartbeat**.

Troubleshooting

See the troubleshooting section of the Configuring WEF for Cribl Stream topic.

16. Destinations

Cribl Stream 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 Summary

Cribl Stream supports the following Destinations. Streaming Destinations accept events in real time. All HTTP-based Destinations are proxyable.

Amazon S3 Compatible Stores

- HTTP/S
- Non-streaming
- Filesystem-based

The Amazon S3 Compatible Stores Destination can be adapted to send data to downstream services like Databricks and Snowflake, for which Cribl Stream currently has no preconfigured Destination. For details, please contact Cribl Support.

Amazon CloudWatch Logs

- HTTP/S
- Streaming

Data Lakes > Amazon S3

- HTTPS only
- Non-streaming

- Filesystem-based

Data Lakes > Amazon Security Lake

- HTTP/S
- Non-streaming
- Filesystem-based

Amazon Kinesis Streams

- HTTP/S
- Streaming

Amazon MSK

- TCP
- Streaming

Amazon SQS

- HTTP/S
- Streaming

Azure Blob Storage

- HTTPS only
- Batching
- Filesystem-based

Azure Data Explorer

- HTTPS only
- Streaming or non-streaming

Azure Event Hubs

- TCP
- Streaming

Azure Monitor Logs

- HTTPS only
- Streaming

Azure Sentinel

- HTTP/S
- Streaming

Confluent Cloud

- TCP
- Streaming

CrowdStrike Falcon LogScale

- HTTPS only
- Streaming

Datadog

- HTTPS only
- Streaming

Elasticsearch

- HTTP/S
- Streaming
- Load-balanced

Elastic Cloud

- HTTPS only
- Streaming
- Load-balanced

Exabeam

- HTTP/S
- Non-streaming
- Filesystem-based

Filesystem/NFS

- Non-streaming
- Filesystem-based

Google Chronicle

- HTTPS only
- Streaming

Google Cloud Logging

- HTTPS only
- Streaming

Google Cloud Pub/Sub

- HTTPS only
- Streaming

Google Cloud Storage

- HTTPS only
- Non-streaming
- Filesystem-based

Grafana Cloud

- HTTP/S
- Streaming

Graphite

- TCP or UDP
- Streaming

Honeycomb

- HTTPS only
- Streaming

InfluxDB

- HTTP/S
- Streaming

Kafka

- TCP

- Streaming

Loki

- HTTP/S
- Streaming

MinIO

- HTTP/S
- Non-streaming
- Filesystem-based

New Relic Events

- HTTPS only
- Streaming

New Relic Logs & Metrics

- HTTPS only
- Streaming

OpenTelemetry (OTel)

- gRPC over HTTP/S or TCP
- Streaming

Prometheus

- HTTP/S
- Streaming

SentinelOne DataSet

- HTTPS only
- Streaming

SignalFx

- HTTPS only
- Streaming

SNMP Trap

- UDP
- Streaming

Splunk HEC

- HTTP/S
- Streaming
- Load-balanced

Splunk Load Balanced

- TCP
- Streaming
- Load-balanced

Splunk Single Instance

- TCP
- Streaming

StatsD

- TCP or UDP
- Streaming

StatsD Extended

- TCP or UDP
- Streaming

Sumo Logic

- HTTP/S
- Streaming

Syslog

- TCP or UDP
- Streaming
- Load-balanced (TCP only)

TCP JSON

- TCP
- Streaming
- Load-balanced

Wavefront

- HTTPS only
- Streaming

Webhook

- HTTP/S
- Streaming

Internal and Special-Purpose Destinations

These special-purpose Destinations route data within your Cribl Stream deployment, or among Workers across distributed or hybrid Cribl.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 Stream, so it requires no configuration. Useful for testing.
- Cribl HTTP: Send data among peer Worker Nodes over HTTP. Streaming and load-balanced.
- Cribl TCP: Send data among peer Worker Nodes over TCP. Streaming and load-balanced.
- 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 Stream 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 Stream 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).

In Cribl.Cloud, using a staging directory is only available on hybrid, customer-managed Workers.

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 Stream 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.)

For further information about backpressure, see Destination Backpressure Triggers.

You can configure your desired behavior through a Destination's **Backpressure Behavior** drop-down. Where other options are not displayed, Cribl 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, in single-instance deployments, select **Manage**, then proceed to the options below. In distributed deployments, first click **Manage**, then select a **Worker Group** to configure and choose one of the options below.

- To access the graphical QuickConnect UI, click **Destinations**. Next, select the desired type, and then click either **Add New** or (if displayed) Select **Existing**.

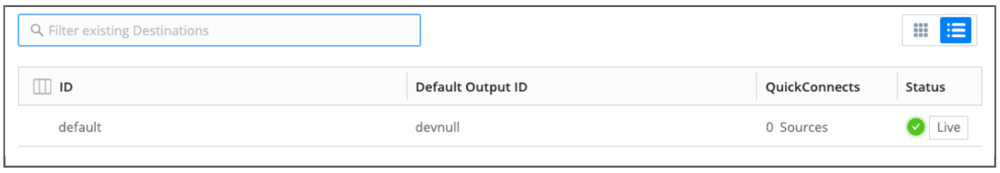
- To access the Routing UI, click **Data** > **Destinations**. On the resulting **Manage Destinations** page's tiles, select the desired type, then click **Add New**.

To edit any Destination's definition in a JSON text editor, click **Manage as JSON** at the bottom of the **New Destination** modal, or on the **Configure** tab when editing an existing Destination. You can directly edit multiple values, and you can use the **Import** and **Export** buttons to copy and modify existing Destination configurations as `.json` files.

When JSON configuration contains sensitive information, it is redacted during export.

Capturing Outgoing Data

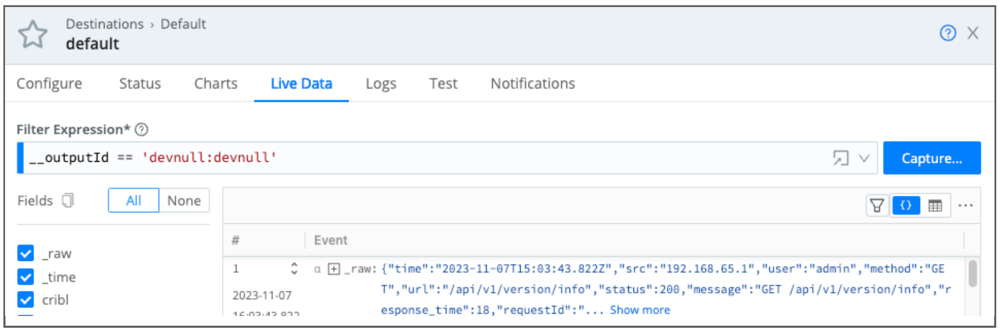
To capture data from a single enabled Destination, you can bypass the Preview pane, and instead capture directly from a **Manage Destinations** page. Just click the **Live** button beside the Destination you want to capture.



ID	Default Output ID	QuickConnects	Status
default	devnull	0 Sources	✓ Live

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.



```
__outputId == 'devnull:devnull'
```

```
#    Event
1    _raw: {"time":"2023-11-07T15:03:43.822Z","src":"192.168.65.1","user":"admin","method":"GE
2023-11-07  T","url":"/api/v1/version/info","status":200,"message":"GET /api/v1/version/info","r
            esponse_time":18,"requestId":"... Show more
```

Destination modal > Live Data tab

Monitoring Destination Status

Each Destinations's configuration modal offers two tabs for monitoring: **Status** and **Charts**.

Status Tab

The **Status** tab provides details about the Workers in the group and their status. An icon shows whether the Worker is operating normally.

You can click each Worker's row to see specific information, for example, to identify issues when the Destination displays an error. The specific set of information provided depends on the Destination type. The data represents only process 0 for each Worker Node.

> The content of the **Status** tab is loaded live when you open it and only displayed when all the data is ready. With a lot of busy Workers in a group, or Workers located far from the Leader, there may be a delay before you see any information.
>
> The statistics presented are reset when the Worker restarts.

Charts Tab

The **Charts** tab presents a visualization of the recent activity on the Destination. The following data is available:

- Events in
- Thruput in (events per second)
- Bytes in
- Thruput in (bytes per second)
- Blocked status

> This data (in contrast with the status tab) is read almost instantly and does not reset when restarting a Worker.

16.1. Amazon

16.1.1. Amazon CloudWatch Logs

Cribl Stream supports sending data to Amazon CloudWatch Logs. Cribl Stream 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 Stream to Output to Amazon CloudWatch Logs

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Authentication

Use the **Authentication method** drop-down 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 Stream 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for CloudWatch Logs: Toggle to `Yes` to use Assume Role credentials to access CloudWatch Logs.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

16.1.2. Amazon Kinesis Streams

Cribl Stream can output events to **Amazon Kinesis Data Streams** records of up to 1MB uncompressed. Cribl Stream does **not** have to run on AWS in order to deliver data to a Kinesis Data Stream.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Meeting the Needs of Different Downstream Receivers

You can use this Destination to send data to different kinds of downstream receivers that have different expectations. In all cases what the Destination sends out are Kinesis Records. This works because the Amazon Kinesis Data Streams Service is flexible about what is actually in a Record, as the two following use cases show.

Sending Multiple Events within One Record

When used to feed data into Amazon Kinesis Data Firehose, this Destination will start with multiple events as NDJSON, gzip-compress them together into a blob, and send that as a single Kinesis Record. This uses the Kinesis PutRecord API call to send the combined events in a single Record.

Firehose will gunzip-uncompress the Record and split the NDJSON out into individual events again. There are also other downstream receivers besides Firehose that can do the same thing.

This approach requires either:

- **Advanced Settings** > **Compression** set to `Gzip`, **or**
- **Advanced Settings** > **Send batched** toggled on.

Meanwhile, if you want Cribl Stream to be the consumer, on the Kinesis Streams Source, set **Record data format** to `Cribl`. See the format details below.

Sending One Event Per Record

In contrast to the first use case, some downstream receivers require a Kinesis Record to contain exactly one Cribl event. Here, this Destination uses the Kinesis PutRecords API call to send multiple events in a single API call, but where each event has a Record to itself. (Note the difference between `PutRecord` in the previous use case and `PutRecords` in this one.)

This approach requires:

- **Advanced Settings** > **Compression** set to `None`, **and**
- **Advanced Settings** > **Send batched** toggled off.

Meanwhile, if you want Cribl Stream to be the consumer, on the Kinesis Streams Source, set **Record data format** to `Newline JSON`. See the format details below.

Configuring Cribl Stream to Output to Amazon Kinesis Data Streams

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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** drops the newest events from being sent out of Cribl Stream, and throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Authentication

Use the **Authentication Method** drop-down 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 Stream 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.

For details, see AWS' Actions, resources, and condition keys for Amazon Kinesis Data Streams documentation.

Manual: If not running on AWS, you can select this option to enter a static set of user-associated IAM credentials (your access key and secret key) directly or by reference. This is useful for Workers not in an AWS VPC, 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for Kinesis Streams: Toggle to `Yes` to use Assume Role credentials to access Kinesis Streams.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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`.

Compression: Type of compression to use on records being sent downstream. Defaults to `Gzip`, and when set to `None`, displays the following control:

Send batched: Whether to send multiple events batched as a single NDJSON Kinesis record (the default) or send each event as its own Kinesis record, serialized as JSON. When toggled off, displays the following control:

- **Max records per flush**: Maximum number of records to send in a single request to the Amazon Kinesis Data Streams Service PutRecords API endpoint.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Amazon Kinesis Data Streams Permissions

The following permissions are needed to write to an Amazon Kinesis Stream:

- `kinesis:DescribeStream` - needed in all cases
- `kinesis:PutRecord` - needed only when **Advanced Settings** > **Compression** is set to `Gzip` or **Advanced Settings** > **Send batched** is toggled on.
- `kinesis:PutRecords` - needed only when **Advanced Settings** > **Send batched** is toggled off.

Record Formats in Detail

When sending multiple events within one record, the Record format will be as follows:

- A header line provides information about the payload (the one supported type is shown below).
- Each Kinesis record will contain multiple events serialized as Newline-Delimited JSON (NDJSON).
- Record payloads (including header and body) will be gzip-compressed if **Advanced Settings** > **Compression** is set to `Gzip`.

Sample Kinesis Record

```
{"format":"ndjson","count":8,"size":3960}
{"_raw":"07-03-2018 18:33:51.136 -0700 ERROR TcpOutputFd - Read error. Connection r
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to 127.0.0.
...
```

When sending one event per Record, the Record format will be as follows:

- No header will be included.
- Each Kinesis record will contain a single event serialized as JSON (**not** NDJSON).
- Record payloads will not be compressed.

16.1.3. Amazon S3 Compatible Stores

S3 is a non-streaming Destination type. Cribl Stream 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 Stream currently has no preconfigured Destination. For these integrations, please contact Cribl Support.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cribl Stream to Output to S3 Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down 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 Stream 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for S3: Toggle to `Yes` to use Assume Role credentials to access S3.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.
- In Cribl Edge 4.3 and later, the S3 Collector supports ingesting data in the Parquet format. Therefore, data that you export in Parquet format can be replayed via the Collector.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When this limit is exceeded, on any individual Worker Process, Cribl Stream will close the oldest open files, and move them to the final output location. Defaults to `100`.

Cribl Stream will close files when **any** of the four above conditions is met.

Staging file limit: Maximum number of files that the Destination will allow to wait for upload before it applies backpressure. Defaults to `100`; minimum is `10`; maximum is `4200`. For context, see Troubleshooting below.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

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 Stream version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Stream 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 Stream 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 Stream will automatically add it to the URL path. For details, see AWS' Path-Style Requests topic.

Object ACL: Object ACL (Access Control List) to assign to uploaded objects.

Storage class: Select a storage class for uploaded objects. Defaults to `Standard`. The other options are: `Reduced Redundancy Storage`; `Standard, Infrequent Access`; `One Zone, Infrequent Access`; `Intelligent Tiering`; `Glacier Flexible Retrieval`; `Glacier Instant Retrieval`; or `Glacier Deep Archive`.

Server-side encryption: Encryption type for uploaded objects – used to enable encryption on data at rest. Defaults to no encryption; the other options are `Amazon S3 Managed Key` or `AWS KMS Managed Key`.

> AWS S3 always encrypts data in transit using HTTPS, with default one-way authentication from server to clients. With other S3-compatible stores (such as our native MinIO Destination), use an `https://` URL to invoke in-transit encryption. Two-way authentication is not required to get encryption, and requires clients to possess a certificate.

Signature version: Signature version to use for signing S3 requests. Defaults to `v4`.

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (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`. For context, see Troubleshooting below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are always needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`

- `kms:Decrypt`

See the AWS documentation.

Temporary Access via SSO Provider

You can use Okta or SAML to provide access to S3 buckets using temporary security credentials.

Internal Fields

Cribl Stream 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

Misconfigured or Nonexistent Buckets

If the S3 bucket that the Destination is trying to send files to does not exist or is misconfigured, the Destination logs will reflect the following pattern:

1. Upon initialization, the Destination will log an error that says the bucket cannot be found. This error will not be logged if **Advanced Settings** > **Verify if bucket exists** is turned off.
2. Every time the Destination "rolls" a new file out of staging to try to upload it to S3, the Destination will log an error, until the number of files rolled exceeds the value of **Advanced Settings** > **Max files to stage**.

Once the Destination has begun to apply backpressure:

- The Destination will log a backpressure warning.
- Events will stop flowing through the Destination.
- The Destination will stop creating staging files.
- Although the Destination will periodically attempt to upload the staged files, it will not log any additional failures.

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.

16.1.4. Amazon SQS

Cribl Stream supports sending events to Amazon Simple Queuing Service.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Send Data to Amazon SQS

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Authentication

Use the **Authentication method** drop-down 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 Stream 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 Stream’s secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for SQS: Toggle to `Yes` to use Assume Role credentials to access SQS.

AWS account ID: Enter the SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account where this Cribl Stream 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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 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.

Fields for this Destination:

- `__messageGroupId`
- `__sqsMsgAttrs`

- `__sqsSysAttrs`

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

16.2. Azure

16.2.1. Azure Blob Storage

Cribl Stream 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 Stream is running on Azure, another cloud platform, or on-prem.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cribl Stream to Output to Azure Blob Storage

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker 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** drop-down 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 Stream 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=<y
```

A fictitious example, using Microsoft's recommended HTTPS option, is:

```
DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32;E
```

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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

> If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Default: `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Default: `100`.

Cribl Stream will close files when **either** of the `Max file size (MB)` or the `Max file open time (sec)` conditions are met.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send one part at a time – that is, to upload the file's contents sequentially. Defaults to `1`; highest allowed value is `10`.

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 Stream version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Stream 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 Stream 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.

16.2.2. Azure Data Explorer

Cribl Stream supports sending data to the Azure Data Explorer (ADX) managed data analytics service; you can then run Kusto queries against the data. This Destination can deliver data to Azure whether Cribl Stream is running on Azure, another cloud platform, or on-prem.

ADX stores log data in databases, which in turn contain one or more of the tables defined in the Azure namespace. You must create a separate Cribl Stream ADX Destination for each table that will store your data.

Type: Non-Streaming **or** Streaming (configurable) | TLS Support: Yes | PQ Support: No

Prerequisites

Before you configure the Destination, you need to collect some information from your Azure deployment. Start from **Home** (`portal.azure.com`), then navigate to the locations described below and collect the items specified.

Information About Your ADX Cluster

From **Home**, navigate to:

Azure Data Explorer Clusters > `your_Azure_cluster_name` > **Overview** > **Essentials**

- Copy the **URI**. In Cribl Stream, this will be the value for **General Settings** > **Cluster base URI**. Make a copy of the URI with `/.default` appended – this will be the value of **Authentication Settings** > **Scope**.
- Copy the **Data Ingestion URI**. In Cribl Stream, this will be the value for **General Settings** > **Ingestion service URI**.

Information About Your Azure Database

From **Home**, navigate to:

Azure Data Explorer Clusters > `your_Azure_cluster_name` | **Databases** > `database_name` | **Query**

- Note the name of your desired target table – this table name will be the value of **General Settings** > **Table name**.

Information About Your Azure App

First, from **Home**, navigate to:

App registrations > `your_Azure_app_name` | **Overview** | **Essentials**

- Copy the **Application (client) ID** – this will be the value of **Authentication Settings** > **Client ID**.
- Copy the **Directory (tenant) ID** – this will be the value of **Authentication Settings** > **Tenant ID**.

Nex, from **Home**, navigate to:

App registrations > `your_Azure_app_name` | **Certificates & secrets**

Now what you do depends on which method for **Authentication** you plan to use in Cribl Stream.

To use the Client secret method

- In the **Client secrets** tab, click **New client secret** and copy the **Value** of the new secret. In Cribl Stream, this will be the value of **Client secret**.

To use the Client secret (text secret) method

- In the **Client secrets** tab, click **New client secret** and copy the **Value** of the new secret. In Cribl Stream, this will be the value of **Create new secret** > **Value**.

To use the Certificate method

After you have generated a certificate and keys (if you do this later on in the configuration process, return to this section then):

- In the **Certificates** tab, click **Upload Certificate**. In the resulting drawer, follow the prompts to upload your certificate, and click **Add** to close the drawer.

The chosen certificate will then appear in the **Certificates** tab's list.

Data Mapping Prerequisites

Finally, you might need to note or copy data mapping information to obtain values for some of the **General Settings**.

ADX uses a data mapping to map fields from incoming data to fields in the target table. To see the data mapping, you'll run a Kusto query against your target table.

Here's the query syntax:

```
.show table EntityName ingestion MappingKind mapping MappingName
```

Here's an example query:

```
.show table SyslogTable ingestion json mapping "SyslogMapping"
```

The Kusto query that originally created this mapping would look like this:

```
.create table SyslogTable ingestion json mapping "SyslogMapping"
```

In Cribl Stream, you can use the names of mappings like the above example in the **General Settings** > **Data mapping** field. Alternatively, you can create a mapping without naming it, then copy and paste it (as a JSON object) into the Cribl Stream UI. To do this, toggle **General Settings** > **Mapping object** on. This hides the default **Data mapping** text field and opens a text window that expects JSON.

Configuring Cribl Stream to Output to Azure Data Explorer

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Azure Data Explorer**. 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** > **Azure Data Explorer**. Next, click **Add Destination** to open a **New Destination** modal that provides the options below.

General Settings

This Destination supports two **ingestion modes**:

- **Batching** (the default) stages data in a storage container from which ADX pulls batches of data, then writes them to the target ADX table. Batching works well when the Destination needs to ingest large amounts of data in short periods of time.
- **Streaming** sends data as HTTP request payloads, directly to the target ADX table, without staging them in a container first. Streaming can achieve lower latency than batching, as long as the data that

the Destination ingests arrives in relatively small amounts. When in this mode, the Retries section (left tab) becomes available.

Output ID: Enter a unique name to identify this Destination definition.

Ingestion mode: Use the buttons to select **Batching** mode (the default) or **Streaming** mode.

Cluster base URI: Enter the base URI for your ADX cluster.

Ingestion service URI: Displayed only in **Batching** mode. Enter the Ingestion Service URI for your ADX cluster.

Database name: Enter the name of the ADX database where the target table resides.

Table name: Enter the name of the target table.

Validate database settings: When you save or start the Destination, validates the database name and credentials that you have entered in these settings. Also validates the table name, except when **Add mapping object** (below) is on. Defaults to `Yes`. Disable if your Azure app does not have **both** the **Database Viewer** and the **Table Viewer** role.

Add mapping object: Displayed only in **Batching** mode, this control is toggled off by default. Toggle on when you want to paste in a data mapping as a JSON object, instead of using a named data mapping. See Data Mapping Prerequisites above.

Data mapping: If **Add mapping object** is off, enter the name of the desired data mapping. If **Add mapping object** is on, enter the desired data mapping as a JSON object.

Compress: By default, this Destination compresses data using `gzip`; otherwise it sends the data uncompressed. This setting is not available when **Data format** is set to `Parquet`.

Data format: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (available only in **Batching** mode, and supported only on Linux, not Windows) exposes a **Parquet Settings** tab, where you select the Parquet schema.

Backpressure behavior: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`. When **Ingestion mode** is set to **Streaming** mode, the **Persistent Queue** option becomes available. See **Persistent Queue Settings** below.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication Settings

Microsoft Entra ID authentication endpoint: From the drop-down, select the Microsoft Entra ID endpoint that provides authentication tokens. To understand the available choices, see the Microsoft Entra documentation.

Tenant ID: Directory ID (tenant identifier) in Microsoft Entra ID.

Client ID: `client_id` to pass in the OAuth request parameter.

Scope: Scope to pass in the OAuth request parameter.

Use the **Authentication method** buttons to select one of these options:

- **Client secret**: Use this default option to enter the client secret that you generated for your app in the Azure portal.
- **Client secret (text secret)**: Use this option to select (or create) a stored text secret.
- **Certificate**: Use this option to select the certificate whose public key you register (or will register) for your app in the Azure portal. Or, create a new one.

Persistent Queue Settings

This tab is displayed when **General Settings** > **Ingestion mode** is set to **Streaming** mode and **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Worker Nodes (with an Enterprise plan), this tab should expose only the destructive **Clear Persistent Queue** button (described below in this section).

Due to Known Issue CRIBL-21335, the on-prem Persistent Queue settings are still displayed in Cribl.Cloud – but you should ignore all controls other than the **Clear Persistent Queue** button.

The other controls are not needed because Cribl.Cloud automatically allocates a maximum queue size of 1 GB disk space per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.

- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: The target memory size for page segments. Generally, lower values improve reading speed, while higher values improve compression. Must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Retries

This tab is displayed when **General Settings** > **Ingestion mode** is set to **Streaming** mode.

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, `429 (Too Many Requests)` is the only response code configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

The controls on this tab vary depending on the **Ingestion mode** selected on the **General Settings** tab. Below you'll find one dedicated section for the Batching, and another for the Streaming controls.

Only the **Environment** control, at the bottom of the tab, is common to both:

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Advanced Settings for Batching Mode

Flush immediately: Toggle on to bypass the data management service's aggregation mechanism. See the ADX documentation.

Retain blob on success: By default, Cribl Stream deletes each data blob once ingestion is complete. Toggle this on if you prefer to retain blobs instead.

Extent tags: Optionally, add strings or tags to partitions of the target table – Azure calls these extents or data shards.

Enforce uniqueness via tag values: Use the **Add value** button to specify a list of `ingest-by` values. Cribl Stream will then check the `ingest-by` tags of incoming extents and discard those whose values match a listed value. This mechanism for avoiding ingesting duplicate extents uses the `ingestIfNotExists` property, as described in the ADX documentation.

Report level: Level for ingestion status reporting. Options are `DoNotReport`, `FailuresOnly` (the default), and `FailuresAndSuccesses`.

Report method: Target for ingestion status reporting. Options are `Queue` (the default), `Table`, and `QueueAndTable`.

Additional fields: Optionally, enter additional configuration properties to send to the ingestion service.

Staging location: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance. Defaults to `/tmp`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks.

Defaults to
`` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

To prevent files from being overwritten, Cribl appends a random sequence of 6 characters to the end of their names. 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.

Max file size (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the storage container. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to the storage container. Defaults to `300` seconds (5 minutes).

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to the storage container. Default: `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to the storage container. Default: `100`.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send one part at a time – that is, to upload the file's contents sequentially. Defaults to `1`; highest allowed value is `10`.

Add output ID: Toggle on if you want Cribl Stream to append the Destination name to staging directory pathnames. This can make it easier to organize and troubleshoot when you have multiple Destinations populating staging directories.

Remove empty staging dirs: When toggled on (the default), Cribl Stream deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

Advanced Settings for Streaming Mode

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Validate server certs: When toggled on (the default) Cribl Stream will reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (such as the system's CA, for example).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Keep alive: By default, Cribl Stream sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Stream to close the connection immediately after sending a request.

Internal Fields

Cribl Stream 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.

16.2.3. Azure Event Hubs

Cribl Stream 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 Stream 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 Stream to Output to Azure Event Hubs

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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`. Setting this control to `Leader` or `All` may significantly slow throughput. If the impact is too great for your use case, set this control to `None`.

Record data format: Format to use to serialize events before writing to the Event Hub Kafka brokers. Defaults to `JSON`.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

TLS Settings (Client Side)

Enabled Defaults to `Yes`.

Validate server certs: Defaults to `Yes`.

Authentication

Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.

Enabled: With the default `Yes` setting, this section's remaining settings are displayed, and all are required settings.

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use with Kafka brokers. Defaults to `PLAIN`, which exposes Basic Authentication options that rely on Azure Event Hubs connection strings. Select `OAUTHBEARER` to enable OAuth Authentication via a different set of options.

Basic Authentication

Selecting the `PLAIN` SASL mechanism provides the options listed in this section.

Username: The username for authentication. For Event Hubs, this should always be `$ConnectionString`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: Use this default option to enter your Event Hubs connection string's primary or secondary key from the Event Hubs workspace. Exposes a **Password** field for this purpose.
- **Secret**: This option exposes a **Password (text secret)** drop-down, in which you can select a stored secret that references an Event Hubs connection string. The secret can reside in Cribl Stream'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=DummyAccessKe
```

OAuth Authentication

Selecting the `OAUTHBEARER` SASL mechanism provides the options listed in this section.

Microsoft Entra ID authentication endpoint: Specifies the Microsoft Entra ID endpoint from which to acquire authentication tokens. Defaults to `https://login.microsoftonline.com`. You can instead select `https://login.microsoftonline.us` or `https://login.partner.microsoftonline.cn`.

Client ID: Enter the `client_id` to pass in the OAuth request parameter.

Tenant identifier: Enter your 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this case, the original creation time might be important to retain in events Cribl Stream sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in UNIX epoch time (either seconds or milliseconds since the UNIX epoch – both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Internal Fields

Cribl Stream 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.

16.2.4. Azure Monitor Logs

Cribl Stream supports sending data to Azure Monitor Logs.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Azure Monitor Logs

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream will add a prefix and suffix around this DNS name, to construct a URI in this format: `https://<Workspace_ID><your_DNS_name>/api/logs?api-version=<API version>.`

Resource ID: Resource ID of the Azure resource to associate the data with. This populates the `_ResourceId` property, and allows the data to be included in resource-centric queries. (Optional, but if this field is not specified, the data will not be included in resource-centric queries.)

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If keepalives are not supported by the server (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl 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 destination cluster nodes.

16.2.5. Azure Sentinel

Cribl Stream can send log and metric events to the Azure Sentinel SIEM. This Destination encrypts and sends events via the HTTPS protocol. (These docs use the terms "Azure Sentinel" and "Microsoft Sentinel" interchangeably.)

Type: Streaming | TLS Support: No | PQ Support: Yes

Prerequisites

Before configuring your Azure Sentinel Destination(s), you have to prepare the Azure workspace, create data collection rules (DCRs), and obtain the ingestion URL that will receive the data from your Destination(s).

Complete these preparatory steps, which are described in Azure Sentinel Integration, before configuring the Azure Sentinel Destination. That topic explains the overall Sentinel/Cribl workflow, provides necessary context, and shows you how to obtain the values you'll need to configure the Destination.

Behind the scenes, this Destination uses the Azure Monitor Logs Ingestion API.

Configuring Cribl Stream to Output to Azure Sentinel

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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.

If you intend to send multiple event types, or tables, to Sentinel, you'll need a separate Destination for each event type. In this case, Cribl recommends you use a single URL to ingest all the data, with an Output Router to route each event type to the correct Azure Sentinel Destination.

General Settings

Output ID: Enter a unique name to identify this HTTP output definition. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Endpoint configuration: Method for configuring the endpoint. Options include:

- **URL** – lets you directly enter the data collection endpoint. This method is the simplest way of configuring the endpoint. See Obtaining a URL for more information.
- **ID** – lets you enter individual IDs that Cribl Stream uses to create the URL used as the data collection endpoint.

Selecting `URL` exposes the following field.

URL: Endpoint URL to send events to. The internal field `__url`, where present in events, will override the `URL` and `ID` values. See Internal Fields below.

Selecting `ID` exposes the following fields.

Data collection endpoint: Data collection endpoint (DCE) in the format `https://<endpoint-name>.<identifier>.<region>.ingest.monitor.azure.com`.

Data collection rule ID: Immutable ID for the data collection rule (DCR).

Stream name: Name of the Sentinel table in which to store events.

Optional Settings

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication Settings

- **Login URL**: Endpoint for the OAuth API call, which is expected to be a POST call. This URL takes the form `https://login.microsoftonline.com/<tenant-id>/oauth2/v2.0/token`.
- **OAuth secret**: Secret parameter value to pass in the API call's request body.
- **Client ID**: JavaScript expression used to compute the application (client) ID for the Azure application. Can be a constant.

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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a trusted CA (e.g., the system's CA). Toggle this to `No` if you want Cribl Stream 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 Stream to cycle through them in the order returned.

Compress: By default, Cribl Stream enables gzip-compress to compress the payload body before sending. Toggle this off if you want Cribl Stream not to gzip-compress the payload body.

Keep alive: By default, Cribl Stream sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Stream to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30.`

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `1000` KB. Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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 Stream 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 Stream will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Stream 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" }`.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, it will throw away the connection and attempt a new one. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), Cribl Stream will establish a new connection for the next request.

- When resolving the Destination’s hostname, 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 among destination cluster nodes.

16.3. Data Lakes

16.3.1. Cribl Lake Destination

The **Cribl Lake** Destination sends data to the Cribl Lake and automatically selects a partitioning scheme that works well with Cribl Search.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

The Cribl Lake Destination is available only in Cribl.Cloud.

Configure a Cribl Lake Destination

1. From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:
    - To configure via the graphical QuickConnect UI:
        1. Click **Routing**, then **QuickConnect**.
        2. Click **Add Destination** at right.
        3. From the resulting drawer's tiles, select **Data Lakes**, then **Cribl Lake**.
        4. Click either **Add Destination** or (if displayed) **Select Existing**.
    - Or, to configure via the Routing UI:
        1. Click **Data**, then **Destinations**.
        2. From the resulting page's tiles or the **Destinations** left nav, select **Data Lakes**, then **Cribl Lake**.
        3. Click **Add Destination** to open a **New Destination** modal.
2. In the Destination modal, configure the following under **General Settings**:
    - **Output ID**: Enter a unique name to identify this Cribl Lake Destination.
    - **Lake dataset**: Select Cribl Lake dataset to send data to.

    You can't target the built-in `cribl_logs` and `cribl_metrics` datasets with this Destination.
3. Next, you can configure the following **Optional Settings** that you'll find across many Cribl Destinations:

- **Backpressure behavior**: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

4. Optionally, configure any Post-Processing settings outlined in the below sections.
5. Click **Save**, then **Commit & Deploy**.
6. Verify that data is searchable in Cribl Lake.

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 `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

16.3.2. Amazon S3

The **Data Lake Amazon S3 Destination** sends data to Amazon Simple Storage Service (**Amazon S3**) and uses a partitioning scheme that is designed to work with Cribl Search. You can organize your bucket structure by event fields for referencing in Cribl Search. See the Cribl Search setup for Data Lake Amazon S3 for details.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cribl Stream to Output to S3 Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI. Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Data Lakes** > **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**. From the resulting page's tiles or the **Destinations** left nav, select **Data Lakes** > **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}`.

Optional Settings

Partition by fields: List of fields to partition the path by, in addition to time, which is included automatically.

The partitioning scheme is designed to work with Cribl Search. This scheme optimizes how objects are searched relative to the search's time range.

In AWS S3, bucket content is returned in ascending alphanumeric order, by default, they are sorted "oldest-first". This Destination uses alpha characters to prefix objects, so ascending sort returns them in "newest-first" order, allowing Search to find files that will likely have the requested objects soonest.

The effective partition will be `<4-letters>-YYYY/<2-letters>-MM/<2-letters>-DD/<2-letters>-HH/<list/of/fields>`. For example, `hjhh-2023/ag-07/ai-25/aj-14/<field>/`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-downs 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 Stream 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for S3: Toggle to `Yes` to use Assume Role credentials to access S3.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

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 Stream will close the oldest open files, and move them to the final output location. Defaults to `100`.

Cribl Stream will close files when **any** of the four above conditions is met.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Add Output ID: When set to `Yes` (the default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

For a Destination originally configured in a Cribl Stream version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Stream 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).

Endpoint: S3 service endpoint. If empty, Cribl Stream 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 Stream 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`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

IAM Permissions

The following permissions are always needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Internal Fields

Cribl Stream 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.

16.3.3. 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 Stream 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 Stream 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 Stream to Output to Amazon Security Lake

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 Stream Worker 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 Stream.

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 Worker Node Process, Cribl Stream will close the oldest open files, and move them to the final output location. Defaults to `100`.

Cribl Stream will close files when **any** of the four above conditions is met.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

Add Output ID: When set to `Yes` (the default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

Remove staging dirs: Toggle 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.

IAM Permissions

The following permissions are always needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Creating and Modifying Parquet Schemas

When you need to add a new Parquet schema or modify an existing one:

1. Navigate to the Parquet Schema Library, located at **Processing** > **Knowledge** > **Parquet Schemas**.
2. Create or modify the schema, as described here.

Cribl strongly recommends that to modify an existing schema, you first clone it and give the clone its own distinct name.

3. Return to this Destination, and select the new or modified schema in the **Parquet schema** drop-down.
4. Click **Save**.

Cloning First is Safer Than Just Modifying

Modifying an existing schema in the Schema Library does **not** propagate your modifications to the Destination's copy of the schema.

- Cloning and renaming schemas is the safest approach, because in Step 3 above, it removes any doubt that your Destination will now use the newly-modified version of your schema.
- If you do **not** clone and rename the schema (i.e., you leave the schema name unchanged), you still must re-select the schema in the Destination's **Parquet schema** drop-down, and click **Save**, to bring the modified version into the Destination.

Internal Fields

Cribl Stream 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.

16.4. Elastic

16.4.1. Elasticsearch

Cribl Stream can send events to an Elasticsearch cluster using the Bulk API. As of v.3.3, Cribl Stream supports Elastic data streams.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output to Elasticsearch

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Load balancing: When enabled (default), lets you specify multiple bulk API URLs and load weights. With the `No` setting, if you notice that Cribl Stream 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

Use the **Bulk API URLs** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add URL**. Each row provides the following fields:

URL: Specify the URL to an Elastic node to send events to – e.g., `http://elastic:9200/_bulk`

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 toggle appears when **Load balancing** is set to `Yes`. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Type: Specify document type to use for events. An event's `__type` field can overwrite this value.

Backpressure behavior: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream will append `/<worker-id>/<output-id>` to this field's value.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the `Block` option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Click this “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (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. (This option is visible only when the **General Settings** > **Load balancing** option is set to `No`.)

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Extra parameters: Name-value pairs to pass as additional parameters. If you are using Elastic ingest pipelines, specify an extra parameter whose name is `pipeline` and whose value is the name of your pipeline, similar to these examples.

Elastic version: Determines how to format events. For Elastic Cloud, you must explicitly set version `7.x`. For other Elasticsearch clusters, the `Auto` default will discover the downstream Elasticsearch version automatically, but you have the option to explicitly set version `6.x` or `7.x`.

Elastic pipeline: To send data to an Elastic Ingest pipeline, optionally enter that pipeline's name as a constant. Or, enter a JavaScript expression that evaluates outgoing events and sends matching events to the desired Elastic Ingest pipeline(s). Cribl Stream will first interpret your entry as a constant and try to find a matching value in the event. If it finds no matching value, it will evaluate your **Elastic pipeline** entry as an expression.

For example, the expression `sourcetype=='access_common'?'cribl_pipeline':undefined` matches events whose `sourcetype` is `access_common`, and sends them to an Elastic Ingest pipeline named `cribl_pipeline`.

You can also specify the name of the pipeline in an event field. For example, `myPipelineField` would use the value from the event's `myPipelineField` property (if present) to identify the Elastic Ingest pipeline to process the event. Alternately, the expression `myPipelineField != null ? myPipelineField : undefined` would also identify this field. If the event does not contain such a field, `myPipelineField != null ? myPipelineField : 'theDefaultIndex'` will provide a default index. With this approach, you can override the Elastic Ingest pipeline at the event level.

See also the Elasticsearch Source documentation.

The next two fields appear only when the **General Settings** > **Load balancing** option is set to `Yes`.

DNS resolution period (seconds): Re-resolve any hostnames each time this interval recurs, and pick up destinations from the A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds.

Include document_id: Toggle this setting to `No` to omit the `document_id` field when sending events to an Elastic TSDS (time series data stream).

Write action: Action to use when writing events. Set this option to `Create` (default) when writing to a data stream, which is append-only. Set this option to `Index` only when you write directly to an index and need to update existing records. `Index` will fail if you use it to write to a data stream.

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).
- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Field Normalization

This Destination normalizes the following fields:

- `_time` becomes `@timestamp` at millisecond resolultion.
- `host.name` is set to `host`.

See also our Elasticsearch Source documentation's **Field Normalization** section.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__id`
- `__type`
- `__index`
- `__host`

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname with load balancing disabled, Cribl 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 Elasticsearch cluster nodes.

16.4.2. Elastic Cloud

Cribl Stream can send events to Elastic Cloud.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Elastic Cloud

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

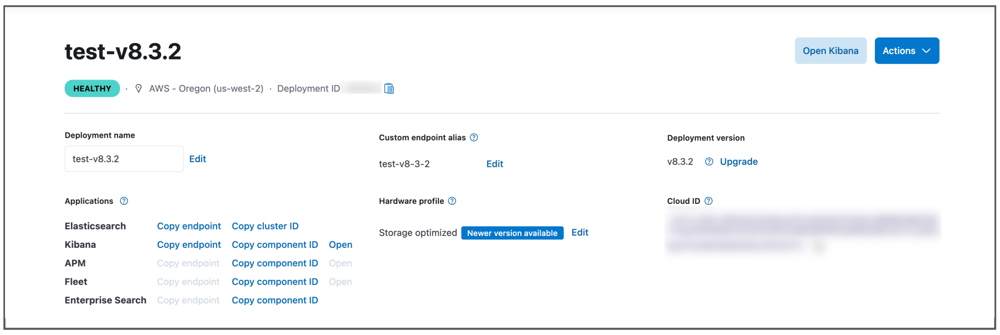
To configure via the graphical QuickConnect UI, click **Routing** > **QuickConnect** (Stream) or **Collect** (Edge). Next, click **Add Destination** at right. From the resulting drawer's tiles, select **Elastic Cloud**. Next, click either **Add Destination** or (if displayed) **Select Existing**. The 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 **Elastic 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 Elastic Cloud Destination definition.

Cloud ID: Enter the Cloud ID of the Elastic Cloud environment where you want to send events. You'll find it on the Deployments overview page of your Elastic Cloud environment.



test-v8.3.2

AWS - Oregon (us-west-2) · Deployment ID [redacted]

Deployment name: test-v8.3.2	Custom endpoint alias: test-v8-3-2	Deployment version: v8.3.2
Applications: Elasticsearch, Kibana, APM, Fleet, Enterprise Search	Hardware profile: Storage optimized	Cloud ID: [redacted]

Getting your Elastic Cloud ID

Data stream or index: Enter a JavaScript expression that evaluates to the name of the Elastic data stream or Elastic index where you want events to go. The expression is evaluated for each event, can evaluate to a constant value, and must be enclosed in quotes or backticks. An event's `__index` field can overwrite the index or data stream name.

Elastic pipeline: Enter the name of the Elastic ingest pipeline (optional). You can use an expression to override the pipeline attribute received from the Elasticsearch Source.

Authentication Settings

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

Backpressure behavior: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed 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 PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process.

Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior.

This setting is required and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. Cribl Stream will append `/<worker-id>/<output-id>` to this field's value.

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 Stream 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 **Group 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.)

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 Stream Node that processed the event.

- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or

3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Extra parameters: Name-value pairs to pass as additional parameters. If you are using Elastic ingest pipelines, specify an extra parameter whose name is `pipeline` and whose value is the name of your pipeline, similar to these examples.

Include document_id: Toggle this setting to `No` to omit the `document_id` field when sending events to an Elastic TSDS (time series data stream).

Failed request logging mode: Determines which data is logged when a request fails. Use the drop-down to select one of these options:

- `None` (default).
- `Payload`.
- `Payload + Headers`. Use the **Safe Headers** field below to specify the headers to log. If you leave that field empty, all headers are redacted, even with this setting.

Safe headers: List the headers you want to log, in plain text.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Field Normalization

This Destination normalizes the following fields:

- `_time` becomes `@timestamp` at millisecond resolultion.
- `host.name` is set to `host`.

See also our Elasticsearch Source documentation's **Field Normalization** section.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__id`
- `__type`
- `__index`
- `__host`

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

16.5. Google Cloud

16.5.1. Google Chronicle

Cribl Stream 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 Stream 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 Stream to Output to Chronicle

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream 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 Stream 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Use the **Authentication method** drop-down 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream to automatically add to events using this output. Wildcards are supported. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, `429 (Too Many Requests)` and `503 (Service Unavailable)` are the only response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `30000` (30 seconds).

- **Backoff multiplier**: The base for the exponential backoff algorithm; defaults to `1`. A value of `2` means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Defaults to `30000` (30 seconds); minimum is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream 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): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: 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 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.

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.

16.5.2. Google Cloud Logging

Cribl Stream 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 Stream to Output to Google Cloud Logging

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Auto**: This option uses the `GOOGLE_APPLICATION_CREDENTIALS` environment variable, and requires no configuration here.
- **Manual**: This default option displays a **Service account credentials** field for you to enter the contents of your service account credentials file (a set of JSON keys), as downloaded from Google Cloud.
  To insert the file itself, click the upload button at this field's upper right. As an alternative, you can use environment variables, as outlined here.
- **Secret**: This option exposes a drop-down in which you can select a stored secret that references the service account credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

This tab 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Advanced Settings

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Connection timeout: Amount of time (in milliseconds) to wait for the connection to establish before retrying. Defaults to `10000` (10 sec.).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30` sec.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

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`

16.5.3. Google Cloud Pub/Sub

Cribl Stream 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 Stream to Output to Pub/Sub

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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

When working with Google Cloud Pub/Sub, you need to know:

- The project where your credentials originated.
- The project that contains any topic you want to work with.

If the desired topic resides in the same project where your credentials originated, describe it using either its short name **or** its fully-qualified name.

If the desired topic resides in a different project than the one where your credentials originated, you **must** describe it using its fully-qualified name.

Here's an example of a fully-qualified topic name: `projects/my-project-id/topics/my-topic-id`.

The short name for the same topic would be: `my-topic-id`.

Output ID: Enter a unique name to identify this Pub/Sub output definition. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Topic ID: ID of the Pub/Sub topic to send events to. This static initial configuration of the topic is required; but you can optionally also override it dynamically on a per-event basis.

Optional Settings

Create topic: Toggle to `Yes` if you want Cribl Stream to create the topic on Pub/Sub if it does not exist.

Ordered delivery: Toggle to `Yes` if you want Cribl Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk

> space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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 Stream 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 Stream's **Create topic** option, your service account should have one of the following (or higher) roles:

- `roles/pubsub.editor`
- `roles/editor`

Either `editor` role confers multiple permissions, including those from the lower `viewer`, `subscriber`, and `publisher` roles. For additional details, see the Google Cloud Access Control topic.

Proxying Requests

If you need to proxy HTTP/S requests, see System Proxy Configuration.

Let's Change the Topic

The Pub/Sub Destination supports alternate topics specified at the event level in the `__topicOut` field. So (for example) if a Pub/Sub Destination is configured to send to main topic `topic1`, and Cribl Stream receives an event with `__topicOut: topic2`, then Cribl Stream 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 Stream cannot dynamically create the topic, and will drop the event. On the Destination's **Status** tab, the **Dropped** metric tracks the number of events dropped because a specified alternate topic did not exist.

(This example uses the short topic name, and that works as long as the topic resides in the project where your GCP credentials originated. If the topic resides in a different project, you must use the fully-qualified topic name, for example `__topicOut: projects/<my-project-id>/topics/topic2`.)

16.5.4. 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 Stream to send data to Google Cloud Storage buckets, ideally set the following access permissions on the Cloud Storage side:

- Fine-grained access control must be enabled on the buckets.
- The Google service account or user should have the Storage Admin or Owner role.

For narrower access than the above roles, the Google account should have the following access to buckets:

- `roles/storage.objectCreator` role.
- `roles/storage.objectViewer` role.
- If the `Verify if bucket exists` advanced option is enabled, one bucket-level `storage.buckets.list` permission.

For details, see the Cloud Storage Overview of Access Control and Understanding Roles documentation.

Configuring Cribl Stream to Output to Cloud Storage Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **Auto**: This option authenticates with the attached Google Cloud Platform (GCP) Service Account, relying on GCP IAM roles to access the appropriate GCP resources. This option is available only when Cribl Stream is on-prem, and the Worker Nodes are running in Google Compute Engine VMs on GCP.
- **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 Stream Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.

- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

> If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

Cribl Stream 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 Stream 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.

16.6. Kafka

16.6.1. Kafka

Cribl Stream 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 Stream must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Stream to Output to Kafka

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.

Compression: Codec to compress the data before sending to Kafka. Select `None`, `Gzip` `Snappy`, or `LZ4`.

Cribl strongly recommends enabling compression. Doing so improves Cribl Stream's performance, enabling faster data transfer using less bandwidth.

Backpressure behavior: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Stream 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): Leave this field blank. See Connecting to Kafka below.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

SASL mechanism: Use this drop-down to select the SASL authentication mechanism to use. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these SASL mechanisms, select one of the following buttons:

Manual: Displays **Username** and **Password** fields to enter your Kafka credentials directly.

Secret: This option exposes a **Credentials secret** drop-down in which you can select a stored text secret that references your Kafka credentials. A **Create** link is available to store a new, reusable secret.

GSSAPI/Kerberos

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the keytab file for the authentication principal.

Principal: Enter the authentication principal, e.g.: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, e.g.: `kafka`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. See Kafka Authentication with Kerberos for further detail.

Schema Registry

This section governs Kafka Schema Registry Authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Data Sources documentation:

- A **trace** tracks the progression of a single request.
  - Each trace is a tree of **spans**.
  - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
  - A metric contains individual measurements called **data points**.
- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Stream logs the error.

Connecting to Kafka

Leave the TLS Settings > Server name (SNI) field blank

In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

> Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.
2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service – among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later.

In this case, the original creation time might be important to retain in events Cribl Stream sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in UNIX epoch time (either seconds or milliseconds since the UNIX epoch – both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

16.6.2. Confluent Cloud

Cribl Stream 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 Stream 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 **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.

Compression: Codec to use to compress the data before sending to Kafka. Select `None`, `Gzip`, `Snappy`, or `LZ4`. Defaults to `Gzip`.

Backpressure behavior: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 Stream'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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Stream 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): Leave this field blank. See Connecting to Kafka below.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server’s cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section documents the SASL (Simple Authentication and Security Layer) authentication settings to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

- **SASL mechanism**: Select the SASL (Simple Authentication and Security Layer) authentication mechanism to use. Defaults to `PLAIN`. `SCRAM-SHA-256`, `SCRAM-SHA-512`, and `GSSAPI/Kerberos` are also available. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these SASL mechanisms, select one of the following buttons:

Manual: Displays **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 keytab 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`.

You will also need to set up your environment and configure the Cribl Stream host for use with Kerberos. See Kafka Authentication with Kerberos for further detail.

Schema Registry

This section governs Kafka Schema Registry Authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: Defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (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 Stream.

- **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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

Authentication timeout (ms): Maximum time to wait for Kafka to respond to an authentication request. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Data Sources documentation:

- A **trace** tracks the progression of a single request.
  - Each trace is a tree of **spans**.
  - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
  - A metric contains individual measurements called **data points**.
- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Stream logs the error.

Connecting to Kafka

Leave the TLS Settings > Server name (SNI) field blank

In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation. This is especially important for Confluent Cloud Dedicated clusters, which rely on SNI – as managed by the Kafka library – for routing.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.
2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service – among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

16.6.3. Amazon MSK

Cribl Stream 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 Stream must send events directly to receivers. You might need to adjust your firewall rules to allow this traffic.

Configuring Cribl Stream to Output to Kafka

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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`. When set to `Protobuf`, the **Protobuf Format Settings** section (left tab) becomes available.

Compression: Codec to compress the data before sending to Kafka. Select `None`, `Gzip`, `Snappy`, or `LZ4`. Defaults to `Gzip`.

Cribl strongly recommends enabling compression. Doing so improves Cribl Stream's performance, enabling faster data transfer using less bandwidth.

Backpressure behavior: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 Stream 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 **Group Settings** > Worker Processes.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Calculating the Time PQ Will Take to Engage

PQ will not engage until Cribl Stream 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 Stream 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): Leave this field blank. See Connecting to Kafka below.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Authentication

Use the **Authentication method** drop-down to select an AWS authentication method.

Auto: This default option uses the AWS 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 Stream 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 Stream's secrets manager. A **Create** link is available to store a new, reusable secret.

Assume Role

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

Enable for MSK: Toggle on to use Assume Role credentials to access MSK.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming role. This is required only when assuming a role that requires this ID in order to delegate third-party access. For details, see AWS' documentation.

Duration (seconds): Duration of the Assumed Role's session, in seconds. Minimum is 900 (15 minutes). Maximum is 43200 (12 hours). Defaults to 3600 (1 hour).

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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. Defaults to `1000` (1 second).

Reauthentication threshold (ms): If the broker requires periodic reauthentication, this setting defines how long before the reauthentication timeout Cribl Stream initiates the reauthentication. Defaults to `10000` (10 seconds).

A small value for this setting, combined with high network latency, might prevent the Destination from reauthenticating before the Kafka broker closes the connection.

A large value might cause the Destination to send reauthentication messages too soon, wasting bandwidth.

The Kafka setting `connections.max.reauth.ms` controls the reuthentication threshold on the Kafka side.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Protobuf Format Settings

Definition set: From the drop-down, select `OpenTelemetry`. This makes the **Object type** setting available.

Object type: From the drop-down, select `Logs`, `Metrics`, or `Traces`.

Working with Protobufs

This Destination supports supports Binary Protobuf payload encoding. The Protobufs it sends can encode traces, metrics, or logs, the three types of telemetry data defined in the OpenTelemetry Project's Data Sources documentation:

- A **trace** tracks the progression of a single request.
  - Each trace is a tree of **spans**.
  - A span object represents the work being done by the individual services, or components, involved in a request as that request flows through a system.
- A **metric** provides aggreggated statistical information.
  - A metric contains individual measurements called **data points**.

- A **log**, in OpenTelemetry terms, is "any data that is not part of a distributed trace or a metric".

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry/proto/trace/v1/trace.proto.
- For metrics, opentelemetry/proto/metrics/v1/metrics.proto.
- For logs, opentelemetry/proto/logs/v1/logs.proto.

This Destination will drop non-conforming events. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Stream logs the error.

Connecting to Kafka

Leave the TLS Settings > Server name (SNI) field blank

In Cribl Stream's Kafka-based Sources and Destinations (including this one), the Kafka library that Cribl Stream uses manages SNI (Server Name Indication) without any input from Cribl Stream. Therefore, you should leave the **TLS Settings** > **Server name (SNI)** field blank.

Setting this field in the Cribl Stream UI can cause traffic to be routed to the wrong brokers, because it interferes with the Kafka library's operation.

Connecting to a Kafka cluster entails working with hostnames for **brokers** and **bootstrap servers**.

Brokers are servers that comprise the storage layer in a Kafka cluster. Bootstrap servers handle the initial connection to the Kafka cluster, and then return the list of brokers. A broker list can run into the hundreds. Every Kafka cluster has a `bootstrap.servers` property, defined as either a single `hostname:port` K-V pair, or a list of them. If Cribl Stream tries to connect via one bootstrap server and that fails, Cribl Stream then tries another one on the list.

In the **General Settings** > **Brokers** list, you can enter either the hostnames of brokers that your Kafka server has been configured to use, or, the hostnames of one or more bootstrap servers. If Kafka returns a list of brokers that's longer than the list you entered, Cribl Stream keeps the full list internally. Cribl Stream neither saves the list nor makes it available in the UI. The connection process simply starts at the beginning whenever the Source or Destination is started.

Here's an overview of the connection process:

1. From the **General Settings** > **Brokers** list – where each broker is listed as a hostname and port – Cribl Stream takes a hostname and resolves it to an IP address.

2. Cribl Stream makes a connection to that IP address. Notwithstanding the fact that Cribl Stream resolved **one** particular hostname to that IP address, there may be **many** services running at that IP address – each with its own distinct hostname.
3. Cribl Stream establishes TLS security for the connection.

Although SNI is managed by the Kafka library rather than in the Cribl Stream UI, you might want to know how it fits into the connection process. The purpose of the SNI is to specify one hostname – i.e., service – among many that might be running on a given IP address within a Kafka cluster. Excluding the other services is one way that TLS makes the connection more secure.

Working with Event Timestamps

By default, when an incoming event reaches this Destination, the underlying Kafka JS library adds a `timestamp` field set to the current time at that moment. The library gets the current time from the `Date.now()` JavaScript method. Events that this Destination sends to downstream Kafka receivers include this `timestamp` field.

Some production situations require the `timestamp` value to be the time an incoming event was originally created by an upstream sender, not the current time of its arrival at this Destination. For example, suppose the events were originally created over a wide time range, and arrive at the Destination hours or days later. In this case, the original creation time might be important to retain in events Cribl Stream sends to downstream Kafka receivers.

Here is how to satisfy such a requirement:

1. In a Pipeline that routes events to this Destination, use an Eval Function to add a field named `__kafkaTime` and write the incoming event's original creation time into that field. The value of `__kafkaTime` **must** be in UNIX epoch time (either seconds or milliseconds since the UNIX epoch – both will work). For context, see this explanation of the related fields `_time` and `__origTime`.
2. This Destination will recognize the `__kafkaTime` field and write its value into the `timestamp` field. (This is similar to the way you can use the `__topicOut` field to overwrite a topic setting, as described above.)

If the `__kafkaTime` field is **not** present, the Destination will apply the default behavior (using `Date.now()`) described previously.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Fields for this Destination:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

16.7. Metrics

16.7.1. Graphite

Cribl Stream supports sending data to a Graphite backend Destination.

Type: Streaming | TLS Support: No | PQ Support: Yes

Configuring Cribl Stream to Output to a Graphite Backend

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

16.7.2. StatsD

Cribl Stream supports sending data to a StatsD Destination.

Type: Streaming | TLS Support: No | PQ Support: Yes

Configuring Cribl Stream to Output via StatsD

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 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 Stream, 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

16.7.3. StatsD Extended

Cribl Stream'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 Stream to Output via StatsD Extended

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 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 Stream, 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

16.8. New Relic Ingest

16.8.1. New Relic Events

Cribl Stream 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 Stream's New Relic Logs & Metrics Destination.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output Events to New Relic

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Defaults to `Yes`, meaning compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. Defaults to `0`, allowing unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name**/**Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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 Stream-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 Stream.

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.

16.8.2. New Relic Logs & Metrics

Cribl Stream 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 Stream 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 Stream 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 Stream's performance and data flow by installing New Relic's Cribl dashboard.

Configuring Cribl Stream to Output to New Relic

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other

options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low 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 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.

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 Stream-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.

16.9. Prometheus

16.9.1. Prometheus

Cribl Stream 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 Stream to Output to Prometheus

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Node that processed the event) and `cribl_wp` (Cribl Stream Worker Process that processed the event). Supports wildcards. Other options include:

- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (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.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Metric renaming expression: A JavaScript expression that can be used to rename metrics.
The default expression – `name.replace(/\\./g, \'_\')` – replaces all `.` characters in a metric's name with the Prometheus-supported `_` character. Use the `name` global variable to access the metric's name. You can access event fields' values via `__e.<fieldName>`.

Send metadata: Whether to generate and send metrics' metadata (`type` and `metricFamilyName`) along with the metrics. The default `Yes` value displays this additional field:

- **Metadata flush period (sec)**: How frequently metrics metadata is sent out. Value must at least equal the base **Flush period (sec)**. (In other words, metadata cannot be flushed on a shorter interval.) Defaults to `60` seconds.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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 Stream uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Stream will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Stream 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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl 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 Prometheus cluster nodes.

16.9.2. Grafana Cloud

Cribl Stream 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 Stream

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 Stream 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 Stream to Output to Grafana Cloud

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk

> space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.
>
> This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream. 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 Stream Node that processed the event) and `cribl_wp` (Cribl Stream 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 Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (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.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Loki and Prometheus might complain about entries being delivered out of order when **Request concurrency** is set > `1` and any of **Flush period (sec)**, **Max body size (KB)**, or **Max events per request** are set to low values.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Metric renaming expression: A JavaScript expression that can be used to rename metrics.
The default expression – `name.replace(/\\./g, \'_\')` – replaces all `.` characters in a metric's name with the Prometheus-supported `_` character. Use the `name` global variable to access the metric's name. You can access event fields' values via `__e.<fieldName>`.

Message format: Whether to send events as `Protobuf` (the default) or `JSON`.

Compress: When the **Message format** is `JSON`, this setting controls the data compression format used before sending the data to Grafana Cloud. Defaults to `Yes` for GZIP-compression. (Applies only to Loki's JSON payloads. This toggle is hidden when the **Message format** is `Protobuf`, because both Prometheus' and Loki's Protobuf implementations are Snappy-compressed by default.)

Logs message field: The event field to send as log output, for example: `_raw`. All other event fields are discarded. If left blank, Cribl Stream 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 Stream uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Stream will send it Prometheus as a metric event. If `__criblMetrics` is absent, Cribl Stream 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 Stream sends the value from `__labels` to Loki. Setting the `__labels` field in a Pipeline gives you a quick way to experiment with the logs being sent.

If there are no labels set (this would happen when **System fields**, **Logs labels**, and `__labels` are all empty), Cribl Stream 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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination’s hostname, Cribl 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 Grafana Cloud nodes.

16.9.3. Loki

Cribl Stream can send log events to Grafana's Loki log aggregation system.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Loki

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Node that processed the event) and `cribl_wp` (Cribl Stream 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 Stream Source that processed the event.

- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream 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 before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `15.`

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Message format: Whether to send events as `Protobuf` (the default) or `JSON.`

Compress: When the **Message format** is `JSON`, this setting controls the data compression format used before sending the data to Loki. Defaults to `Yes` for GZIP-compression. (Applies only to Loki's JSON payloads.

This toggle is hidden when the **Message format** is `Protobuf`, because both Prometheus' and Loki's Protobuf implementations are Snappy-compressed by default.)

Logs message field: The event field to send as log output, for example: `_raw`. All other event fields are discarded. If left blank, Cribl Stream 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.

16.10. Splunk

16.10.1. 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 Stream to Output to Splunk HEC Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream 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

Use the **Splunk HEC Endpoints** table to specify a known set of receivers on which to load-balance data. It appears only when **Load balancing** is toggled to `Yes`.

To specify more receivers on new rows, click **Add Endpoint**. Each row provides the following fields:

HEC Endpoint: Specify the URL to a Splunk HEC endpoint to send events to – e.g., `http://localhost:8088/services/collector/event`.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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** drop-down 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
response: {"text":"Data channel is missing","code":10}
time: 2023-02-14T15:26:03.413Z
```

Optional Settings

Exclude current host IPs: This toggle appears when **Load balancing** is set to `Yes`. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Output multi-metrics: Toggle to `Yes` to output multiple-measurement metric data points. (Supported in Splunk 8.0 and above, this format enables sending multiple metrics in a single event, improving the efficiency of your Splunk capacity.)

Validate server certs: Reject certificates that are not authorized by a trusted CA (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. (This setting is available only when **General Settings** > **Load balancing** is set to `No`.)

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Max body size (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

> If you're sending data to Splunk Cloud, Cribl recommends a maximum value of `1 MB` for **Max body size (KB)**. This upper limit is defined by the maximum content length for the default HTTP Event Collector (HEC) provided by Splunk Cloud.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1`.

- Retries happen on this flush interval.
- Any HTTP response code in the `2xx` range is considered success.
- Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
- Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **Add Header** to add **Name/Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

The next two fields take effect only in the Cribl App for Splunk. (Cribl Stream 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 Stream 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 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.

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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives – or if the server closes a pooled connection while idle – a new connection will be established for the next request.
- When resolving the Destination's hostname with load balancing disabled, Cribl Stream will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Splunk HEC servers.
- See Splunk's documentation on editing `fields.conf` to ensure the visibility of index-time fields sent to Splunk by Cribl Stream.

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.

16.10.2. 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 Stream's Splunk Load Balanced Destination.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output to Splunk Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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** drops the newest events from being sent out of Cribl Stream, and throws away incoming data, while leaving the contents of the PQ unchanged.

Clear Persistent Queue: Click this "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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 Stream 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 Stream 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 Stream 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.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the payloads of failed requests. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.

4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Stream is storing payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

Notes about Forwarding to Splunk

- Data sent to Splunk is not compressed.
- The only `ack` from indexers that Cribl Stream listens for and acts upon is the shutdown signal described in Minimize in-flight data loss above.
- If events have a Cribl Stream 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 Stream.

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.

16.10.3. 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 Stream to Load-Balance to Multiple Splunk Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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** drops the newest events being sent out of Cribl Stream, 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream will randomly select this subset of discovered IPs to connect to. Cribl Stream will rotate IPs in future resolution periods – monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Nested field serialization: Specifies whether and how to serialize nested fields into index-time fields. Select `None` (the default) or `JSON`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, enter the shared secret token to use when establishing a connection to a Splunk indexer.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token described above. A **Create** link is available to store a new, reusable secret.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads to Troubleshoot Closed Connections below.

Endpoint health fluctuation time allowance (ms): How long (in milliseconds) each receiver endpoint can report blocked, before this Destination as a whole reports unhealthy, blocking senders. (Grace period for transient fluctuations.) Use `0` to disable the allowance; default is `100` ms; maximum allowed value is `60000` ms (1 minute).

Throttling: Throttle rate, in bytes per second. Multiple byte units such as KB, MB, GB, 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.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the payloads of failed requests. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Stream is storing payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

SSL Configuration for Splunk Cloud – 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 Stream 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-splu
```

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 Stream'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>_serv
```

If you are using RSA keys, instead use:

```
openssl rsa -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_ser
```

Step 4. Extract the public key for the Server Certificate:

```
openssl x509 -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-cloud>_se
```

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 Stream listens for and acts upon is the shutdown signal described in Minimize in-flight data loss above.
- If events have a Cribl Stream 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 Stream.

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.

16.11. Internal

16.11.1. Cribl HTTP

The Cribl HTTP Destination sends data to a Cribl HTTP Source in the same Distributed deployment, including Cribl.Cloud, at no charge. It's common to send data from Edge and Stream within the same deployment using a Cribl HTTP Destination and Cribl HTTP Source pair.

The Cribl HTTP Destination is available only in Distributed deployments. In single-instance mode, or for testing, you can substitute it with 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 TCP traffic.

How It Works

You can use the Cribl HTTP Destination to transfer data between Workers. If the Cribl HTTP Destination sends data to its Cribl HTTP Source counterpart on another Worker, you're billed for ingress only once – when Cribl first receives the data. All data subsequently relayed to other Workers via a Cribl HTTP Destination/Source pair is not charged.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (on-prem) Node sends data to a Worker in Cribl.Cloud for additional processing and routing to Destinations. However, the Cribl HTTP Destination/Source pair can similarly reduce your metered data ingress in other scenarios, such as on-prem Edge to on-prem Stream.

As one usage example, assume that you want to send data from one Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.
- Create an on-prem Cribl HTTP Destination.
- Create a Cribl HTTP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl HTTP Destination, and from there to the Cribl HTTP Source in Cribl.Cloud.

- On Cribl-managed Cribl.Cloud 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 Node that is connected to the same Leader as the Cribl HTTP Source(s).
- This Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).
- Cribl 3.5.4 was a breakpoint in Cribl HTTP Leader/Worker communications. Nodes running the Cribl HTTP Destination on Cribl 3.5.4 and later can receive data only from Nodes running v.3.5.4 and later. Nodes running the Cribl HTTP Destination on Cribl 3.5.3 and earlier can receive data only from Nodes running v.3.5.3 and earlier.

Configuring a Cribl HTTP Destination

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** toggle displays the following controls.

Exclude Current Host IPs: This toggle determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Cribl Worker Endpoints: In this table, you specify a set of Cribl Workers on which to load-balance data. To specify more Workers on new rows, click **Add Endpoint**. Each row provides the following fields.

- **Cribl Endpoint**: Enter the URL of a Worker to send events to.

  Must point to the **Address** and **Port** configured on a peer Cribl HTTP Source to which you're sending.

- **Load weight**: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (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 before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Exclude fields: Fields to exclude from the event. By default, `__kube_*` and `__metadata` are excluded. This Destination forwards all other Internal Fields.

If you are running Cribl Stream 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.

16.11.2. Cribl TCP

The Cribl TCP Destination sends data to a Cribl TCP Source in the same Distributed deployment, including Cribl.Cloud, at no charge. It's common to send data from Edge and Stream within the same deployment using a Cribl TCP Destination and Cribl TCP Source pair.

The Cribl TCP Destination is available only in Distributed deployments. In single-instance mode or for testing, you can substitute it with the TCP JSON 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 HTTP Destination in certain circumstances, such as when a firewall or proxy allows TCP traffic.

How It Works

You can use the Cribl TCP Destination to transfer data between Workers. If the Cribl TCP Destination sends data to its Cribl TCP Source counterpart on another Worker, you're billed for ingress only once – when Cribl first receives the data. All data subsequently relayed to other Workers via a Cribl TCP Destination/Source pair is not charged.

This use case is common in hybrid Cribl.Cloud deployments, where a customer-managed (on-prem) 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 Node deployed on-prem, to another that is deployed in Cribl.Cloud. You could do the following:

- Create an on-prem File System Collector (or whatever Collector or Source is suitable) for the data you want to send to Cribl.Cloud.
- Create an on-prem Cribl TCP Destination.
- Create a Cribl TCP Source, on the target Stream Worker Group or Edge Fleet in Cribl.Cloud.
- For an on-prem Node configure a File System Collector to send data to the Cribl TCP Destination, and from there to the Cribl TCP Source in Cribl.Cloud.
- On Cribl-managed Cribl.Cloud 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 Node that is connected to the same Leader as the Cribl TCP Source.
- When you configure the Cribl TCP Destination, its **Address** and **Port** values must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Cribl 3.5.4 was a breakpoint in Cribl TCP Leader/Worker communications. Nodes running the Cribl TCP Source on Cribl 3.5.4 and later can send data only to Nodes running v.3.5.4 and later. Nodes running the Cribl TCP Source on Cribl 3.5.3 and earlier can send data only to Nodes running v.3.5.3 and earlier.

Configuring a Cribl TCP Destination

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream will handle excess data by blocking it, dropping it, or queueing it to disk.

Backpressure behavior: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** toggle displays the following controls:

Exclude Current Host IPs

This toggle determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Destinations

The **Destinations** table is where you specify a set of receivers on which to load-balance data.
Click **Add Destination** to specify more receivers on new rows. Each row provides the following fields:

Address: Hostname of a receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on the host.

Each **Address/Port** combination must match the **Address** and **Port** configured on a peer TCP Source to which you're sending.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP); otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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 Stream 4.0.x (or earlier) and are using the Kubernetes Metrics Source with this Destination, consider excluding the following fields to reduce event size:

`!__inputId,!__outputId,!__criblMetrics,__*` .

The contents of `__raw` are often redundant with the `_raw` field's contents. Where they are identical, consider excluding one of the two.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname – i.e., multiple A records – all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl Stream load balancing, IP settings take priority over those from hostnames.)

Max connections: Constrains the number of concurrent receiver connections, per Worker Process, to limit memory utilization. If set to a number > `0`, then on every DNS resolution period, Cribl Stream will randomly select this subset of discovered IPs to connect to. Cribl Stream will rotate IPs in future resolution periods – monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.

- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the payloads of failed requests. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Stream is storing payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

Internal Fields Loopback to Sources

The Cribl TCP and Cribl HTTP Destinations differ from all other Destinations in the way they handle internal fields: They normally send data back to their respective Cribl Sources – where Cribl internal fields, metrics, and sender-generated fields can all be useful.

These Destinations forward all internal fields by default, except for any that you exclude in **Advanced Settings** > **Exclude fields**.

As examples, if the following fields are present on an event forwarded by a Cribl HTTP or Cribl TCP Destination, they'll be accessible in the ingesting Cribl HTTP/TCP Source: `__criblMetrics`, `__srcIpPort`, `__inputId`, and `__outputId`.

16.11.3. Cribl Stream (Deprecated)

This Destination was deprecated as of Cribl Stream 3.5, and has been removed as of v.4.0. Please instead use the Cribl TCP or the Cribl HTTP Destination to enable Worker Nodes to send data to peer Nodes.

The Cribl Stream Destination enables Edge Nodes, and/or Cribl Stream instances, to send data to one or multiple Cribl Stream instances.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Use New Destinations Instead

The replacement Destinations, Cribl TCP and Cribl HTTP, don't rely on IP filtering, for the following reasons:

- Load balancers and/or proxies between the Cribl Destination and Cribl Source can change the IP address, resulting in a bad match and rejected ingest.
- A Lookup table of all IP addresses needed to be sent to each Worker Node/Edge Node from the Leader, which is not scalable.
- The Lookup table of IP addresses required constant communication between the Worker Node/Edge Nodes and the Leader, making this fragile and placing an arbitrary reliance on the Leader that shouldn't be there.

Configuring a Cribl Stream Destination

In the **QuickConnect** UI: Click **Add Destination** beside **Destinations**. From the resulting drawer's tiles, select **Cribl Stream**. Next, click **Add Destination** to open a **New Cribl Stream** drawer.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Destinations**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Destinations**.

From the resulting page's tiles or the **Destinations** left nav, select **Cribl Stream**. Next, click **New Destination** to open a **Cribl Stream** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Address: Hostname of the receiver.

Port: Port number to connect to on the host.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, you can optionally enter an auth token to use in the connection header.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Compression: Codec to use to compress the data before sending. Defaults to `None`.

Throttling: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB etc. are also allowed, e.g., `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Stream will handle excess data by blocking it, dropping it, or queueing it to disk.

Backpressure behavior: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.

- `cribl_wp` – Cribl Stream Worker Process that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream 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.

16.11.4. Default

The **Default** Destination simply enables you to specify a default output from among your already-configured Destinations.

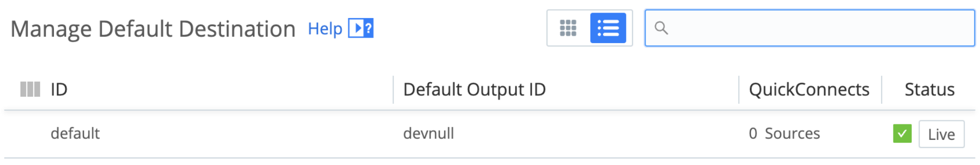
Type: Internal | TLS Support: N/A | PQ Support: N/A

Configuring Cribl Stream's Default Destination

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). A **Default** Destination is preconfigured and ready to use in the right **Destinations** column. Hover over the tile and click its **Configure** button to proceed.

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.



ID	Default Output ID	QuickConnects	Status
default	devnull	0 Sources	✓ Live

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 Stream'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.

16.11.5. 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 Stream to Forward to DevNull

DevNull requires no configuration: A DevNull Destination is preconfigured and active as soon as you install Cribl Stream.

To verify this, from the top nav, click **Manage**, then select a **Worker Group** 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.

16.11.6. 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 Stream to Send to an Output Router

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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
`method=='POST' || method=='GET`	**Seattle**	No
`method=='POST' || method=='GET'`	**Los Angeles**	Yes
`true`	**New York**	Yes

16.12. 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 Stream 3.5.x, this Destination was labeled **Humio HEC**. Some links from this page might still lead to "Humio"-branded resources that CrowdStrike has not renamed.)

Recommendations

To load-balance to customer-managed LogScale receivers, Cribl recommends placing a load balancer in front of cluster nodes, following LogScale Manual Cluster Deployment recommendations. If you are using LogScale Cloud, it provides its own load balancing.

We recommend sending events with the `sourceType` field set to a LogScale parser. This tells LogScale which parser to use to extract fields (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 Stream to Output to CrowdStrike Falcon LogScale Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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** drop-down 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.

- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or

3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Defaults to `Yes` to reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (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 Stream 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 per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`. Each request can potentially hit a different HEC receiver.

Max body size (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1`.

- Retries happen on this flush interval.

> - Any HTTP response code in the `2xx` range is considered success.
> - Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
> - Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **Add Header** to add **Name/Value** pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream 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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives – or if the server closes a pooled connection while idle – a new connection will be established for the next request.
- Cribl 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 CrowdStrike Falcon LogScale servers.

16.13. Datadog

Cribl Stream can send log and metric events to Datadog. (Datadog supports metrics only of type `gauge`, `counter`, and `rate` via its REST API.)

Cribl Stream 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 Stream to Output to Datadog

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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** drop-down 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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`. 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

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.

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 Stream uses a set of internal fields to assist in forwarding data to a Destination.

If an event contains the internal field `__criblMetrics`, Cribl Stream will send it to Datadog as a metric event. Otherwise, Cribl Stream 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

16.14. Exabeam Security Operations Platform

Cribl Stream supports sending data to the Exabeam security operations platform or (as some people think of it) SIEM.

The Exabeam Destination supersedes and improves upon the "old way" to get data from Cribl Stream to Exabeam using the Cribl Webhook Destination.

Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Understanding How Exabeam and Cribl Stream Work Together

Exabeam parsers expect to receive events in original, unmodified form. To make sure this happens, you'll sometimes need to use a combination of Event Breakers and Pipelines. In general, knowing this can help you avoid problems or troubleshoot any that do crop up.

When you send data to Exabeam, Cribl Packs are your friends, because Cribl and Exabeam have partnered to create a set of Exabeam-specific Packs, each of which ensures that events from one common data source arrives at Exabeam in precisely the expected form. Cribl Packs also give you the option of dropping events on a per-data-source basis, so that Cribl Stream never sends irrelevant events to Exabeam. To view the Exabeam-specific Cribl Packs, navigate to the **Manage Packs** page's **New Pack** submenu, select **Add from Dispensary**, then enter `Exabeam` in the filter field.

Cribl strongly recommends that you use **only** the Exabeam-specific, Cribl-authored Packs with this Destination. Other Packs can perform transformations – especially reduction – that prevent the Exabeam parsers from triggering.

When the original data source is Microsoft Active Directory (AD) or LDAP, you'll need to set up an Exabeam Site Collector to perform the initial data capture.

Because Exabeam runs in the Google Cloud Storage (GCS) platform, some setting names include references to GCS.

Finally, Cribl recommends placing Cribl Stream Worker Nodes as close as possible to the sources of the events you want to send to Exabeam. This reduces the length of the path that data must traverse before Cribl Stream optimizes it with compression, reduction, dropping events, and/or redacting sensitive information.

Configuring Cloud Storage Permissions

For Cribl Stream to send data to Exabeam buckets, the following access permissions must be set on the Cloud Storage side:

- Fine-grained access control must be enabled on the buckets.
- The Google service account or user must have the Storage Admin or Owner role.

For details, see the Cloud Storage Overview of Access Control and Understanding Roles documentation.

Creating the Cribl Cloud Collector in Exabeam

Log into the Exabeam Security Operations Platform, and in the **Home** page, click **Cloud Collectors**. In the resulting page, click the **Cribl** tile to see a list of Cribl Collectors (the first time you do this, the list will be empty).

Click **New Collector**, and in the resulting **Collector for Cribl** drawer:

1. Enter a name for your new Cribl Cloud Collector.
2. Optionally, in **Advanced Settings**, configure the **Metadata** value **TIMEZONE**
3. If your desired site already exists, select it from the **SITE** drop-down.
    - If your desired site does not exist yet, skip the drop-down and click the **manage your sites** link underneath. This will take you to a separate UI where you will enter a value for the **Metadata** value **SITENAME**, and Exabeam will generate the associated metadata value **SITE ID**. When you have done this, Exabeam will take you back to the previous UI, and you must then select your newly-created site from the **SITE** drop-down.
4. Click **Install**.

A `Hooray!` confirmation modal will now replace the **Collector for Cribl** drawer. In this modal:

1. Click the copy icon to the right of the encoded **Connection String**. Save this string in a secure and handy location – it is required when configuring the Cribl Stream Exabeam Destination.
2. Click **Go to Overview** to exit the modal.

Although optional, configuring metadata is useful because it allows a single instance of the Exabeam SIEM to differentiate between data from multiple sources. This is relevant for large enterprises whose IT infrastructure comprises many sources whose IP addresses overlap.

Configuring Cribl Stream to Output to Cloud Storage Destinations

From the top nav, click **Manage**, then select a **Worker Group** 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 **Exabeam**. 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 **Exabeam**. 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 Exabeam Destination definition. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Click **Autofill with Exabeam Connection String** to automatically enter the values for the following **Exabeam Settings**.

- **Google Cloud Storage bucket**: Name of the destination bucket – in Exabeam's Cribl Cloud Collector this is called **GCS Bucket Name**. This value can be a constant, or a JavaScript expression that can be evaluated only at init time. For example, you can reference a Global Variable like this: `myBucket-${C.vars.myVar}`.
- **Google Cloud Storage bucket region**: In Exabeam's Cribl Cloud Collector this is called **GCS Bucket Region**. It is the GCS region where the bucket is located.
- **Collector instance ID**: In Exabeam's Cribl Cloud Collector this is called **Instance ID**.
- **Access key**: In Exabeam's Cribl Cloud Collector this is called **Access Key**.
- **Secret**: In Exabeam's Cribl Cloud Collector this is called **Secret Key**.
- **Site name**: In Exabeam's Cribl Cloud Collector this is called **SITENAME**.
- **Site ID**: In Exabeam's Cribl Cloud Collector this is called **SITE ID**.
- **Timezone offset**: In Exabeam's Cribl Cloud Collector this is called **TIMEZONE**.

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 `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Advanced Settings

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300` seconds (5 minutes).

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

Cribl Stream will also close any files larger than 10 MB.

Remove empty staging dirs: When toggled on (the default), deletes empty staging directories after moving files. This prevents the proliferation of orphaned empty directories. When enabled, exposes this additional option:

- **Staging cleanup period**: How often (in seconds) to delete empty directories when **Remove empty staging dirs** is enabled. Defaults to `300` seconds (every 5 minutes). Minimum configurable interval is `10` seconds; maximum is `86400` seconds (every 24 hours).

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) can improve performance.

Reject unauthorized certificates: Whether to accept certificates (such as self-signed certificates, for example) that cannot be verified against a valid Certificate Authority. Defaults to `Yes`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

Troubleshooting

Verify that you are sending the original, unmodified events. Study the applicable Exabeam parsers to make sure you understand exactly what they're doing. If you need to modify events, make sure that when you do, the relevant parsers still trigger.

Nonspecific messages from Google Cloud of the form `Error: failed to close file` can indicate problems with the permissions listed above.

16.15. Filesystem/NFS

Filesystem is a non-streaming Destination type that Cribl Stream can use to output files to a local file system or a network-attached file system (NFS).

Type: Non-Streaming | TLS Support: N/A | PQ Support: N/A

In Cribl.Cloud, the Filesystem Destination is only available on hybrid, customer-managed Worker Nodes.

Configuring Cribl Stream to Output to Filesystem Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker Nodes.

Partitioning expression: JavaScript expression that defines how files are partitioned and organized. Default is date-based. If blank, Cribl Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 Stream Pipeline that processed the event). Supports wildcards. Other

options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

Cribl Stream will close files when **either** of the `Max file size (MB)` or the `Max file open time (sec)` conditions are met.

Header line: If set, this line will be written to the beginning of each output file, followed by a newline character. This can be useful for adding a header row to CSV files.

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 Stream version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is so that upon Cribl Stream 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 Stream 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.

16.16. Honeycomb

Cribl Stream supports sending events to a Honeycomb dataset.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Honeycomb

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Dataset name: Name of the dataset to send events to. (E.g., `iLoveObservabilityDataset`.)

Authentication Settings

Use the **Authentication method** drop-down 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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 Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination’s hostname, Cribl 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 destination cluster nodes.

16.17. InfluxDB

Cribl Stream 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 Stream to Output to InfluxDB

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a trusted CA (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.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass as additional HTTP headers. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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.

16.18. MinIO

MinIO is a non-streaming Destination type, to which Cribl Stream can output objects.

> Type: Non-Streaming | TLS Support: Configurable | PQ Support: No

Configuring Cribl Stream to Output to MinIO Destinations

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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.

The **Staging location** field is not displayed or available on Cribl.Cloud-managed Worker 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 Stream 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 Stream's internal `__partition` field can be populated in multiple ways. The precedence order is: explicit **Partitioning expression** value -> `${host}/${sourcetype}` (default) **Partitioning expression** value -> user-defined `event.__partition`, set with an Eval Function (takes effect only where this **Partitioning expression** field is blank).

Compress: Data compression format used before moving to final destination. Defaults to `gzip` (recommended). This setting is not available when **Data format** is set to `Parquet`.

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to ``` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ```, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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** drop-down 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 Stream 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Parquet Settings

To write out Parquet files, note that:

- On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.
- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Automatic schema: Toggle on to automatically generate a Parquet schema based on the events of each Parquet file that Cribl Stream writes. When toggled off (the default), exposes the following additional field:

- **Parquet schema**: Select a schema from the drop-down.

> If you need to modify a schema or add a new one, follow the instructions in our Parquet Schemas topic. These steps will propagate the freshest schema back to this drop-down.

Parquet version: Determines which data types are supported, and how they are represented. Defaults to `2.6`; `2.4` and `1.0` are also available.

Data page version: Serialization format for data pages. Defaults to `V2`. If your toolchain includes a Parquet reader that does not support `V2`, use `V1`.

Group row limit: The number of rows that every group will contain. The final group can contain a smaller number of rows. Defaults to `10000`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Write statistics: Leave toggled on (the default) if you have Parquet tools configured to view statistics – these profile an entire file in terms of minimum/maximum values within data, numbers of nulls, etc.

Write page indexes: Leave toggled on (the default) if your Parquet reader uses statistics from Page Indexes to enable page skipping. One Page Index contains statistics for one data page.

Write page checksum: Toggle on if you have configured Parquet tools to verify data integrity using the checksums of Parquet pages.

Metadata (optional): The metadata of files the Destination writes will include the properties you add here as key-value pairs. For example, one way to tag events as belonging to the OCSF category for security findings would be to set **Key** to `OCSF Event Class` and **Value** to `2001`.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

Cribl Stream will close files when **either** of the `Max file size (MB)` or the `Max file open time (sec)` conditions is met.

Max concurrent file parts: Maximum number of parts to upload in parallel per file. A value of `1` tells the Destination to send the whole file at once. When set to `2` or above, IAM permissions must include those required for multipart uploads. Defaults to `4`; highest allowed value is `10`.

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 Stream version below 2.4.0, the **Add Output ID** behavior will be switched **off** on the backend, regardless of this toggle's state. This is to avoid losing any files pending in the original staging directory, upon Cribl Stream 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.

IAM Permissions

The following permissions are always needed to write to an Amazon S3-compatible object store:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

If your Destination needs to do multipart uploads to S3, two more permissions are needed:

- `kms:GenerateDataKey`
- `kms:Decrypt`

See the AWS documentation.

Internal Fields

Cribl Stream 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.

16.19. NetFlow

Cribl Stream supports receiving NetFlow v5 data via UDP.

Type: **Push** | TLS Support: **No** | Event Breaker Support: **No**

This Source ingests NetFlow records similarly to how it ingests events from other upstream senders: fields are broken out, and the message header is included with each record. If you prefer to render NetFlow data as metrics, use a pre-processing Pipeline or a Route.

Configuring Cribl Stream to Receive NetFlow Data

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). Next, click **+ Add Source** at left. From the resulting drawer's tiles, select [**Push** >] **NetFlow**. 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** >] **NetFlow**. Next, click **Add Source** to open a **New Source** modal that provides the options below.

Sending large numbers of UDP events per second can cause Cribl.Cloud to drop some of the data. This results from restrictions of the UDP protocol.

To minimize the risk of data loss, deploy a hybrid Stream Worker Group with customer-managed Worker Nodes as close as possible to the UDP sender. Cribl also recommends tuning the OS UDP buffer size.

General Settings

Input ID: Enter a unique name to identify this NetFlow Source definition.

Address: Enter the hostname/IP to listen for NetFlow data. For example: `localhost`, `0.0.0.0`, or `::`.

Port: Enter the port number.

Tags: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. 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 PQ-enabled Source, per Worker Process.
>
> The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue. This limit is not configurable. For configurable queue size, compression, mode, and other options below, use a hybrid Group.
>
> For more about PQ modes, see Always On versus Smart Mode.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Source will stop queueing data and block incoming data. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl Stream will append `/<worker-id>/inputs/<input-id>`.

Compression: Optional codec to compress the persisted data after a file is closed. Defaults to `None`; `Gzip` is also available.

You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (that is, all IPs).

IP denylist regex: Regex matching IP addresses whose messages you want this Source to ignore. Defaults to `^$` (that is, every specific IP address in the list). This takes precedence over the allowlist.

UDP socket buffer size (bytes): Optionally, set the `SO_RCVBUF` socket option for the UDP socket. This value tells the operating system how many bytes can be buffered in the kernel before events are dropped. Leave blank to use the OS default. Minimum: `256`. Maximum: `4294967295`.

It may also be necessary to increase the size of the buffer available to the `SO_RCVBUF` socket option. Consult the documentation for your operating system for a specific procedure.

Setting this value will affect OS memory utilization.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

What Fields to Expect

The NetFlow Source does minimal processing of the incoming UDP messages to remain consistent with the internal Cribl event model. For each UDP message received on the socket, you can expect an event with the following fields. We have organized these fields into categories to make them easier to grasp; the categories are our own, and not from the NetFlow specifications. The field definitions are mostly copied from Cisco NetFlow documentation.

General

- `_time`: The UNIX timestamp (in seconds) at which the message was received by Cribl Stream.
- `source`: A string in the form `udp|<remote IP address>|<remote port>`, indicating the remote sender.
- `host`: The hostname of the machine running Cribl Stream that ingested this event.
- `inputInt`: SNMP index of input interface; always set to zero.
- `outputInt`: SNMP index of output interface.
- `protocol`: IP protocol type (for example, TCP = 6; UDP = 17) of the observed network flow.
- `tcpFlags`: Cumulative logical OR of TCP flags in the observed network flow.
- `tos`: IP type of service; switch sets it to the ToS of the first packet of the flow.
- `icmpType`: Type of the ICMP message. Only present if the protocol is ICMP.
- `icmpCode`: Code of the ICMP message. Only present if the protocol is ICMP.

Because this Source reads input data directly from bytes in a compact format, there is nothing suitable to put in a `_raw` field, and the Source does not add a `_raw` field to events.

Flow Source and Destination

- `srcAddr`: Source IP address; in case of destination-only flows, set to zero.
- `dstAddr`: Destination IP address.

- `nextHop`: IP address of next hop router.
- `srcPort`: TCP/UDP source port number.
- `dstPort`: TCP/UDP destination port number. If this is an ICMP flow, this field is a combination of ICMP type and ICMP code, which are broken out separately as `icmpType` and `icmpCode` fields.
- `srcMask`: Source address prefix mask bits.
- `dstMask`: Destination address prefix mask bits.
- `srcAs`: Autonomous system number of the source, either origin or peer.
- `dstAs`: Autonomous system number of the destination, either origin or peer.

Flow Statistics

- `packets`: Packets in the flow.
- `octets`: Total number of Layer 3 bytes in the packets of the flow.
- `startTime`: System uptime, in milliseconds, at the start of the flow.
- `endTime`: System uptime, in milliseconds, at the time the last packet of the flow was received.
- `durationMs`: `endTime` minus `startTime`.

Header

The following are subfields within the `header` field:

- `count`: Number of flows exported in this flow frame (protocol data unit, or PDU).
- `sysUptimeMs`: Current time in milliseconds since the export device booted
- `unixSecs`: Current seconds since 0000 UTC 1970.
- `unixNsecs`: Residual nanoseconds since 0000 UTC 1970.
- `flowSequence`: Sequence counter of total flows seen.
- `engineType`: Type of flow switching engine.
- `engineId`: ID number of the flow switching engine.
- `samplingMode`: The first two bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling mode.
- `samplingInterval`: The remaining 14 bits of what Cisco NetFlow documentation calls `SAMPLING_INTERVAL`. These bits specify a sampling interval.
- `version`: NetFlow export format version number.

Also, the internal fields listed below will be present.

Internal Fields

Cribl 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.

Fields accessible for this Source:

- `__bytes`
- `__inputId`
- `_time`

UDP Tuning

Incoming UDP traffic is put into a buffer by the Linux kernel. Cribl will drain from that buffer as resources are available. At lower throughput levels, and with plenty of available processes, this isn’t an issue. As you scale up, however, the default size of that buffer may be too small.

You can check the current buffer size with:

```
$ sysctl net.core.rmem_max
```

A typical value of about 200 KB is far too small for an even moderately busy syslog server. You can check the health of UDP with the following command. Check the `packet receive errors` line.

```
$ netstat -su
```

If `packet receive errors` is more than zero, you have lost events, which is a particularly serious problem if the number of errors is increasing rapidly. This means you need to increase your `net.core.rmem_max` setting (see earlier).

You can update the live settings, but you’ll also need to change the boot-time setting so next time you reboot everything is ready to roll.

Live change, setting to 25 MB:

```
$ sysctl -w net.core.rmem_max=26214400
net.core.rmem_max = 26214400
$ sysctl -w net.core.rmem_default=26214400
net.core.rmem_default = 26214400
```

For the permanent settings change, add the following lines to `/etc/sysctl.conf`:

```
net.core.rmem_max=26214400
net.core.rmem_default=26214400
```

We recommend you track a few things related to UDP receiving:

- The `netstat -su` command, watching for errors.
- The **Status** tab in the UDP (Raw) Source. In particular, watch for dropped messages. They could indicate you need a bigger buffer under **Advanced Settings** (default: 1000 events). They could also indicate your Worker is encountering pressure further down the Pipeline.
- Especially if you increase your kernel receive buffer as above, watch your Worker processes' memory usage.

16.20. OpenTelemetry (OTel)

Cribl Stream supports sending events to OpenTelemetry Protocol (OTLP)-compliant targets. (Cribl Stream can receive OTel events through the OTel Source.) Besides native OTel Trace and Metric events, you can send Cribl Stream's Gauge metric events through this OTel Destination.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Protocol and Transport Support

In Cribl Stream 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. If the Destination encounters a parsing error when trying to convert an event to OTLP, it discards the event and Cribl Stream logs the error.

Configuring Cribl Stream to Output to OTel

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 add the Retries section (left tab) and 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a

> hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event).

Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

This tab is displayed when **General Settings** > **Optional Settings** > **Protocol** is set to `HTTP`.

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

This tab's options depend on whether **General Settings** > **Protocol** is set to the `gRPC` or `HTTP` transport. The common settings directly are displayed for both transports.

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 before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` sec.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

Additional Settings for gRPC

When **General Settings** > **Protocol** is set to `gRPC`, the **Advanced Settings** tab adds the following options.

Connection timeout: Amount of time (milliseconds) to wait for the connection to establish before retrying. Defaults to `10000` (10 sec.).

Keep alive time (seconds): How often the sender should ping the peer to keep the connection alive. Defaults to `30`.

Metadata: Extra information to send with each gRPC request. Click **Add Metadata** to add each item as a **Key**-**Value** pair. The **Key** field is arbitrary. The **Value** field is a JavaScript expression that is evaluated just once, when this Destination is initialized. If you pass credentials as metadata, Cribl recommends using C.Secret().

Additional Settings for HTTP

When **General Settings** > **Protocol** is set to `HTTP`, the **Advanced Settings** tab adds the following options.

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (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 Stream to cycle through them in the order returned.

Keep alive: By default, Cribl Stream sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Stream 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.

16.21. SentinelOne DataSet

Cribl Stream can send log events to the SentinelOne/Scalyr DataSet platform via the DataSet API. This Destination sends batches of events, as JSON, to that API's addEvents endpoint.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to DataSet

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 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`. 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, `429 (Too Many Requests)` is the only response code configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.

- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Defaults to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (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 Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: 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 Stream 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.

16.22. SignalFx

Cribl Stream supports sending events to SignalFx.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to SignalFx

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Realm: SignalFx realm name (e.g., `us0`). Required.

Authentication Settings

Use the **Authentication method** drop-down 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.

Status Code	Action
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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.

16.23. SNMP Trap

Cribl Stream supports forwarding of SNMP Traps out.

Type: Streaming | TLS Support: No | PQ Support: No

Configuring Cribl Stream to Forward to SNMP Traps

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

SNMP trap destinations: Each row provides the following fields:

- **Address**: Destination host.
- **Port**: Destination port. Defaults to `162`.

Add Destination: Click to specify additional SNMP trap destinations to forward traps to on new rows.

Optional Settings

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream 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.

16.24. Sumo Logic

Cribl Stream can send logs and metrics to Sumo Logic over HTTP. Sumo Logic offers a Hosted Collector that supports HTTP Sources to receive data over an HTTP POST request.

To send traces to Sumo Logic you can use Cribl's OpenTelemetry Destination pointing to Sumo Logic's OTLP/HTTP Source.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

How Sumo Logic Handles Data

- When an event contains the internal field `__criblMetrics`, it's sent to Sumo Logic as a metric event. Otherwise, it's sent as a log event.
- Data sent to Sumo Logic needs to be UTF-8 encoded and they recommend a data payload have a size, before compression, of 100 KB to 1 MB.
- Sumo Logic may impose throttling and caps on your log ingestion to prevent your account from using On-Demand Capacity, see Sumo Logic's manage ingestion documentation for details.

Prerequisites

In Sumo Logic, create or retrieve an HTTP Logs and Metrics Source's unique endpoint URL. See Sumo Logic's HTTP Logs and Metrics Source documentation for details. You will need the `Manage Collectors` role capability in Sumo Logic.

After creating the Source in Sumo Logic, the URL associated with the Source is displayed. Copy the endpoint URL so you can enter it when you configure the Cribl Stream Sumo Logic Destination.

Configure a Sumo Logic Destination

In Cribl Stream, set up a Sumo Logic Destination.

1. From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:
    - To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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**.

- 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.

2. In the Destination modal, configure the following under **General Settings**:
   - **Output ID**: Enter a unique name to identify this Sumo Logic Destination definition. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.
   - **API URL**: Enter the endpoint URL of the Sumo Logic HTTP Source. This is provided by Sumo Logic after creating the Source, see prerequisites for details. For example, `https://endpoint6.collection.us2.sumologic.com/receiver/v1/http/<long-hash>`.
3. Next, you can configure the following **Optional Settings**:
   - **Custom source name** and **Custom source category** are unique settings to Sumo Logic. These allow you to override the Sumo Logic HTTP Source's **Source Name** and **Source Category** values with HTTP headers. Alternatively, you can define the `__sourceName` and `__sourceCategory` fields on events to assign a custom value at the event level.

   The remaining configurations are Cribl settings that you'll find across many Cribl Destinations.
   - **Backpressure behavior**: Whether to block, drop, or queue events when all receivers are exerting backpressure.
   - **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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.
4. Optionally, configure any Persistent Queue, Processing, Retries, and Advanced settings outlined in the below sections.
5. Click **Save**, then **Commit & Deploy**.
6. Verify that data is searchable in Sumo Logic. See the Verify Data Flow section below.

Persistent Queue Settings

This tab 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

> This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5 GB`. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `1024` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low 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 Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

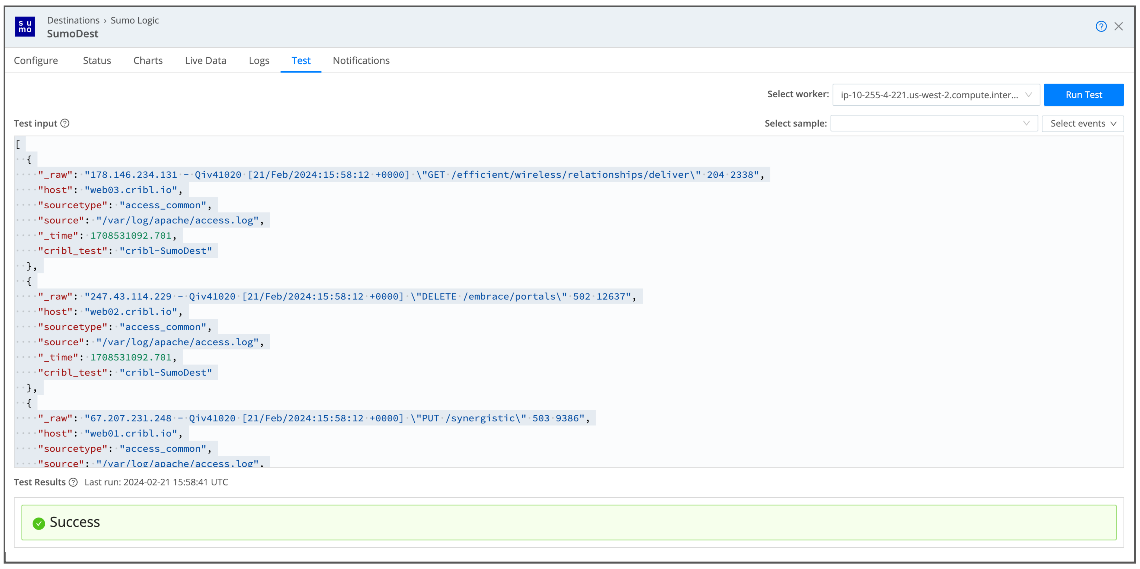
Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Verify Data Flow

To verify data flow, we'll use the Destination's **Test** feature while running a **Live Tail** session in Sumo Logic.

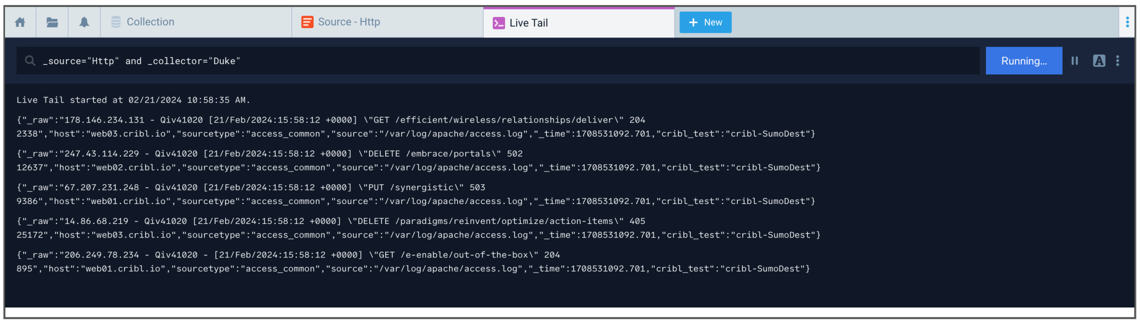
1. In Sumo Logic, run a new Live Tail session against the name of the HTTP Logs and Metrics Source or the **Custom source name** you defined when you configured the Sumo Logic Destination. You'll need to use the `_source` metadata field. For example, if the name of the Source is `HTTP` your search would be `_source="HTTP"`. Ensure you've clicked the **Run** button to start the Live Tail session.
2. In Cribl Stream, open the Destination configuration modal and select the **Test** tab. You can leave the default data or select from the available samples. Click **Run Test**.



```
[
  {
    "_raw": "178.146.234.131 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"GET /efficient/wireless/relationships/deliver\" 204 2338",
    "host": "web03.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
    "_time": 1708531092.701,
    "cribl_test": "cribl-SumoDest"
  },
  {
    "_raw": "247.43.114.229 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"DELETE /embrace/portals\" 502 12637",
    "host": "web02.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
    "_time": 1708531092.701,
    "cribl_test": "cribl-SumoDest"
  },
  {
    "_raw": "67.207.231.248 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"PUT /synergistic\" 503 9386",
    "host": "web01.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
```

Example Destination Test

3. Look back to your Sumo Logic Live Tail session and you should see the sample data from the Cribl test displayed in your Live Tail results.



```
_source="Http" and _collector="Duke"
```

```
Live Tail started at 02/21/2024 10:58:35 AM.
{"_raw":"178.146.234.131 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"GET /efficient/wireless/relationships/deliver\" 204
2338","host":"web03.cribl.io","sourcetype":"access_common","source":"/var/log/apache/access.log","_time":1708531092.701,"cribl_test":"cribl-SumoDest"}
{"_raw":"247.43.114.229 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"DELETE /embrace/portals\" 502
12637","host":"web02.cribl.io","sourcetype":"access_common","source":"/var/log/apache/access.log","_time":1708531092.701,"cribl_test":"cribl-SumoDest"}
{"_raw":"67.207.231.248 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"PUT /synergistic\" 503
9386","host":"web01.cribl.io","sourcetype":"access_common","source":"/var/log/apache/access.log","_time":1708531092.701,"cribl_test":"cribl-SumoDest"}
{"_raw":"14.86.68.219 - Qiv41020 [21/Feb/2024:15:58:12 +0000] \"DELETE /paradigms/reinvent/optimize/action-items\" 405
25172","host":"web03.cribl.io","sourcetype":"access_common","source":"/var/log/apache/access.log","_time":1708531092.701,"cribl_test":"cribl-SumoDest"}
{"_raw":"206.249.78.234 - Qiv41020 - [21/Feb/2024:15:58:12 +0000] \"GET /e-enable/out-of-the-box\" 204
895","host":"web01.cribl.io","sourcetype":"access_common","source":"/var/log/apache/access.log","_time":1708531092.701,"cribl_test":"cribl-SumoDest"}
```

Example Live Tail Results

Troubleshooting

If you receive an error when verifying data flow, take note of the response code and reference Sumo Logic's documentation for common issues to investigate.

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl 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 destination cluster nodes.

16.25. Syslog

Cribl Stream 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 Stream to Output Data in Syslog Format

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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 Stream 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 Stream 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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** toggle replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude current host IPs: This toggle appears when **Load balancing** is set to `Yes`. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Destinations

Use the **Destinations** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add Destination**. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Advanced Settings

The first three options are always displayed:

Octet count framing: When enabled, Cribl Stream prefixes messages with their byte count, regardless of whether the messages are constructed from `message`, `__syslogout`, or `_raw`. When disabled, Cribl Stream omits prefixes, and instead appends a `\n` to messages. RFCs 5425 and 6587 describe how octet counting works in syslog.

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname – i.e., multiple A records – all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl Stream load balancing, IP settings take priority over those from hostnames.)

Max connections: Constrains the number of concurrent receiver connections, per Worker Process, to limit memory utilization. If set to a number > `0`, then on every DNS resolution period, Cribl Stream will randomly select this subset of discovered IPs to connect to. Cribl Stream will rotate IPs in future resolution periods – monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the payloads of failed requests. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.

3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Stream is storing payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

Understanding Syslog Format Options

Unlike other Cribl Stream 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 3164 and/or RFC 5424) before it is transmitted to downstream services.

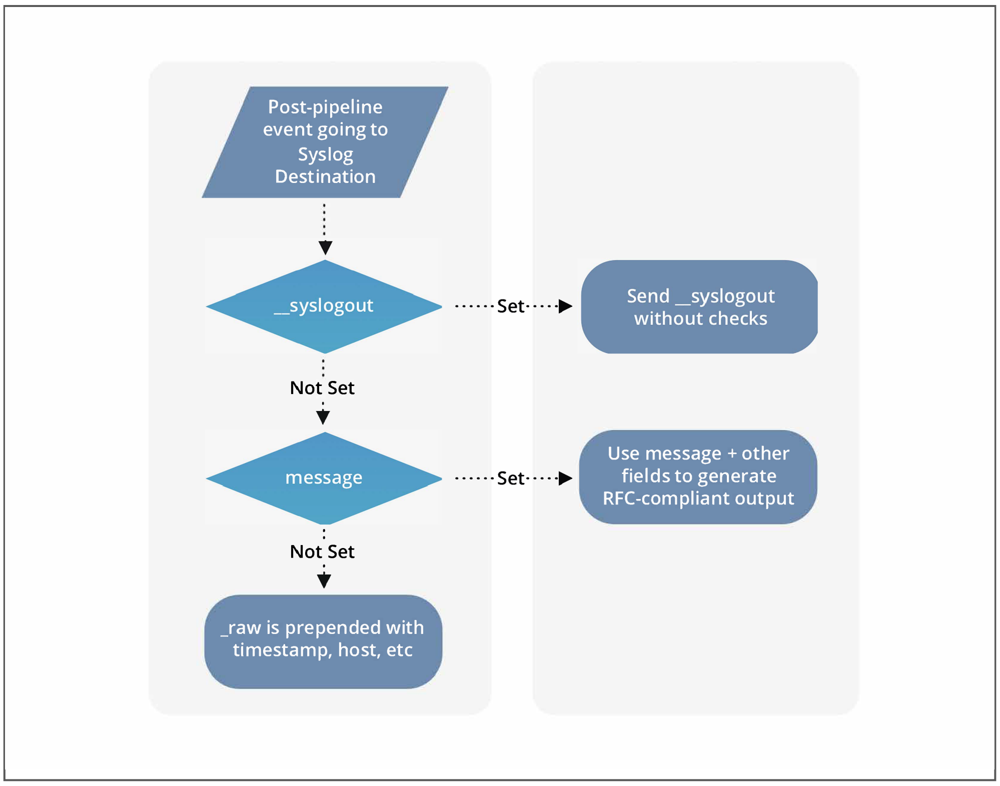
The Syslog Destination's **General Settings** page offers several settings to format the timestamps, to format the message delivering the event, and to set the syslog-specific default settings for Facility, Severity, and App Name.

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 Stream offers three different output approaches in the Syslog Destination. The flowchart below summarizes how Cribl Stream determines which approach to use:

- `message`: For ease of use, Cribl recommends using this option. When you define `message`, Cribl Stream discards `_raw`, and composes a new payload using the other syslog-related fields. Cribl Stream 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 Stream 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 Stream uses the existing `_raw` field as the `message` field, and prepends all the other syslog-related fields to the event.


Post-pipeline event going to Syslog Destination
__syslogout
Set
Send __syslogout without checks
Not Set
message
Set
Use message + other fields to generate RFC-compliant output
Not Set
_raw is prepended with timestamp, host, etc

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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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.

If you have **Octet count framing** enabled for this Destination, Cribl Stream will prepend the number of bytes of the constructed `__syslogout` field to the message before sending it.

The most common uses for the `__syslogout` method are:

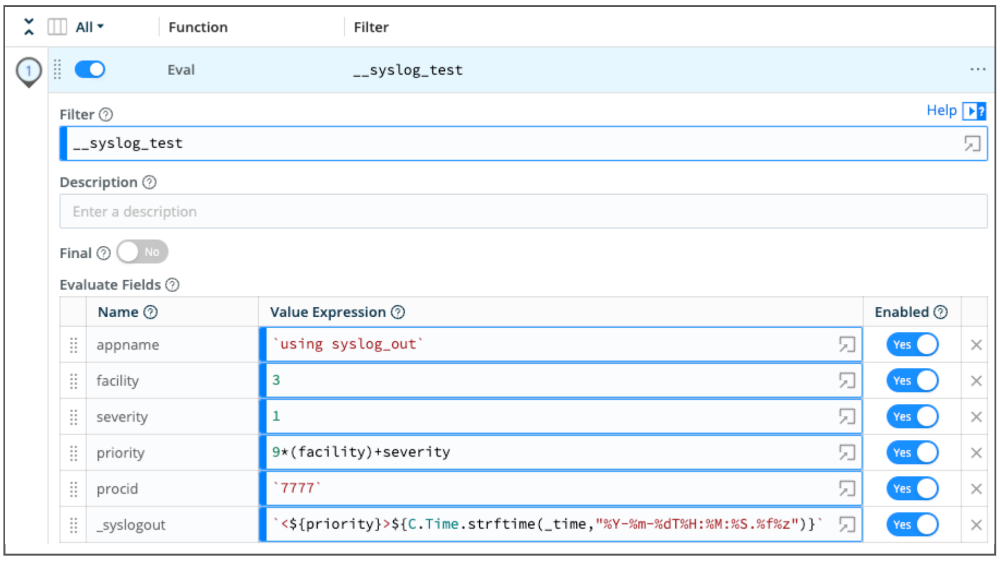
- When the value of `_raw` is already in syslog format as it comes in, and minimal processing is necessary.
- When sending raw TCP data to a destination that doesn't enforce syslog RFCs, such as a raw TCP listener. Cribl Stream currently does not provide a native "raw TCP" Destination. However, in some 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:



Filter: `__syslog_test`

Description: Enter a description

Final: No

Evaluate Fields

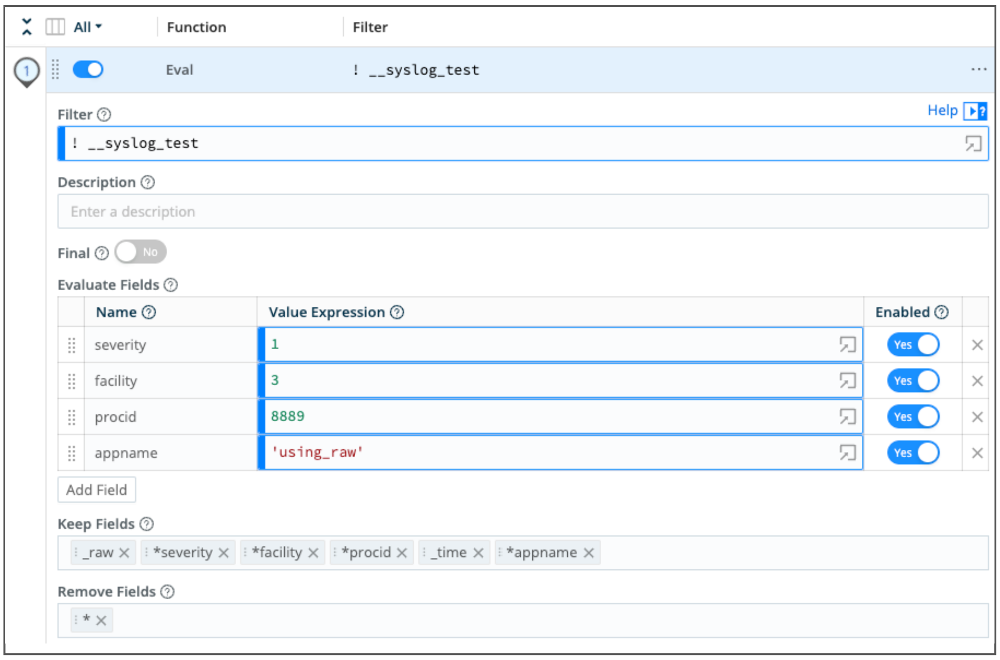
Name	Value Expression	Enabled
appname	`` `using syslog_out` ``	Yes
facility	`3`	Yes
severity	`1`	Yes
priority	`9*(facility)+severity`	Yes
procid	`` `7777` ``	Yes
_syslogout	``` `<${priority}>${C.Time.strftime(_time,"%Y-%m-%dT%H:%M:%S.%f%z")}` ```	Yes

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:



All ▾ | Function | Filter

1 | Eval | `! __syslog_test`

Filter: `! __syslog_test`

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression	Enabled
severity	`1`	Yes
facility	`3`	Yes
procid	`8889`	Yes
appname	`'using_raw'`	Yes

Add Field

Keep Fields: _raw, *severity, *facility, *procid, _time, *appname

Remove Fields: *

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\")} ${hos
            }
          ],
          "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 Stream 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 Stream 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 Stream 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 Stream'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 Stream 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 Stream uses this field to calculate priority. The RFC protocol dictates Facility levels. For details, see Facility.
- `__severity` or **severity**: Cribl Stream also uses this field to calculate priority. The RFC protocol dictates Severity levels. For details, see Severity.
- `__priority`: If you configure this field, Cribl Stream 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 Stream 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 Stream 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 Stream uses the values as specified here:

1. The event contains both `__<key>` and `<key>`: Cribl Stream 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 Stream uses the value of `<key>`. For example, if **host** is set, the value is applied.
3. The event contains neither `__<key>` nor `<key>`, Cribl Stream uses the default values configured in the Syslog Destination.

16.26. TCP JSON

Cribl Stream supports sending data over TCP in JSON format.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output Data in TCP JSON Format

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

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** drop-down 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 toggle appears when **Load balancing** is set to `Yes`. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Compression: Codec to use to compress the data before sending. Defaults to `Gzip`.

Throttling: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB etc. are also allowed, e.g., `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl Stream will handle excess data by blocking it, dropping it, or queueing it to disk.

Backpressure behavior: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** toggle replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Destinations

Use the **Destinations** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add Destination**. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 “panic” button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Advanced Settings

Log failed requests to disk: Toggling to `Yes` makes the payloads of any (future) failed requests available for inspection. See Inspecting Payloads to Troubleshoot Closed Connections below.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Setting General Settings > **Load balancing** to `Yes` adds the following settings:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname – i.e., multiple A records – all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl 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 Stream will randomly select this subset of discovered IPs to connect to. Cribl Stream will rotate IPs in future resolution periods – monitoring weight and historical data, to ensure fair load balancing of events among IPs.

Inspecting Payloads to Troubleshoot Closed Connections

When a downstream receiver closes connections from this Destination (or just stops responding), inspecting the payloads of the resulting failed requests can help you find the cause. For example:

- Suppose you send an event whose size is larger than the downstream receiver can handle.
- Suppose you send an event that has a `number` field, but the value exceeds the highest number that the downstream receiver can handle.

When **Log failed requests to disk** is enabled, you can inspect the payloads of failed requests. Here is how:

1. In the Destination UI, navigate to the **Logs** tab.
2. Find a log entry with a `connection error` message.
3. Expand the log entry.
4. If the message includes the phrase `See payload file for more info`, note the path in the `file` field on the next line.

Now you have the path to the directory where Cribl Stream is storing payloads from failed requests. At the command line, navigate to that directory and inspect any payloads that you think might be relevant.

Format

TCP JSON events are sent in newline-delimited JSON format, consisting of:

1. A header line. Can be empty, 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.

16.27. Wavefront

Cribl Stream supports sending events to Wavefront analytics.

Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Wavefront

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Domain name: WaveFront domain name, e.g., `longboard`. Required.

Authentication Settings

Use the **Authentication method** drop-down 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 to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

On Cribl-managed Cribl.Cloud Workers (with an Enterprise plan), this tab exposes only the destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.

Status Code	Action
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **Add Header** to insert extra headers as **Name/Value** pairs. Values will be sent encrypted.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

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.

16.28. Webhook

Cribl Stream can send log and metric events to webhooks and other generic HTTP endpoints.

Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output via HTTP

From the top nav, click **Manage**, then select a **Worker Group** to configure. Next, you have two options:

To configure via the graphical QuickConnect UI, click Routing > QuickConnect (Stream) or Collect (Edge). 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. If you clone this Destination, Cribl Stream will add `-CLONE` to the original **Output ID**.

Load balancing: When enabled (default), lets you specify multiple Webhook URLs and load weights. With the default `No` setting, if you notice that Cribl Stream is not sending data to all possible IP addresses, enable **Advanced Settings** > **Round-robin DNS**.

Webhook URL: Endpoint URL to send events to.

Webhook URLs

Use the **Webhook URLs** table to specify a known set of receivers on which to load-balance data. To specify more receivers on new rows, click **Add URL**. Each row provides the following fields:

Webhook LB URL: Specify the Webhook URL to send events to – for example, `http://webhook:8000/`

Load weight: Set the relative traffic-handling capability for each connection by assigning a weight (> `0`). This column accepts arbitrary values, but for best results, assign weights in the same order of magnitude to all connections. Cribl Stream 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.

Optional Settings

Exclude current host IPs: This toggle appears when **Load balancing** is set to `Yes`. It determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`, which keeps the current host available for load balancing.

Method: The HTTP verb to use when sending events. Defaults to `POST`. Change this to `PUT` or `PATCH` where required by the endpoint.

Format: The format in which to send out events. One of:

- `NDJSON`: Newline-delimited JSON (the default).
- `JSON Array`: Arrays in JSON-parseable format.
- `Custom`: Events and event batches (payloads) whose format you define using JavaScript **expressions**. See Custom Format below.
- `Advanced`: Events and event batches (payloads) whose format you define using JavaScript **code**. Enables the Webhook Destination to integrate with APIs not otherwise supported in Cribl Stream. See Advanced Format below.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. 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 (for example, `token`); or nested, using a period (for example, `data.token`).
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in requests. Uses `${token}` to reference the token obtained from the login POST request, for example: `` `Bearer ${token}` ``.
- **Refresh interval (secs.)**: How often to refresh the OAuth token. Default `3600` seconds (1 hour); minimum `1` second; maximum `300000` seconds (about 83 hours).
- **OAuth parameters**: Optionally, click **Add parameter** for each additional parameter you want to send in the OAuth login request. Cribl Stream will combine the secret with these parameters, and will send the URL-encoded result in a POST request to the endpoint specified in the **Login URL**. We'll automatically add the `Content-Type` header `application/x-www-form-urlencoded` when sending this request.
  In each row of the resulting table, enter the parameter's **Name**. The corresponding **Value** is a JavaScript expression to compute the parameter's value. This can also evaluate to a constant. Values not formatted as expressions – for example, `id` instead of `` `${id}` `` – will be evaluated as strings.
- **Authentication headers**: Optionally, 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 – for example, `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 destructive **Clear Persistent Queue** button (described below in this section). A maximum queue size of 1 GB disk space is automatically allocated per PQ-enabled Destination, per Worker Process. The 1 GB limit is on outbound uncompressed data, and no compression is applied to the queue.

This limit is not configurable. If the queue fills up, Cribl Stream will block outbound data. To configure the queue size, compression, queue-full fallback behavior, and other options below, use a hybrid Group.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume on each Worker Process. Once this limit is reached, this Destination will stop queueing data and apply the **Queue-full** behavior. Required, and defaults to `5` GB. Accepts positive numbers with units of `KB`, `MB`, `GB`, etc. Can be set as high as `1 TB`, unless you've configured a different **Max PQ size per Worker Process** in Group Settings.

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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group 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 Stream Pipeline that processed the event). Supports wildcards. Other

options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Retries

Honor Retry-After header: Whether to honor a `Retry-After` header, provided that the header specifies a delay no longer than 180 seconds. Cribl Stream limits the delay to 180 seconds even if the `Retry-After` header specifies a longer delay. When enabled, any `Retry-After` header received takes precedence over all other options configured in the **Retries** section. When disabled, all `Retry-After` headers are ignored.

Settings for failed HTTP requests: When you want to automatically retry requests that receive particular HTTP response status codes, use these settings to list those response codes.

For any HTTP response status codes that are not explicitly configured for retries, Cribl Stream applies the following rules:

Status Code	Action
Greater than or equal to `400` and less than or equal to `500`.	Drop the request.
Greater than `500`.	Retry the request.

Upon receiving a response code that's on the list, Cribl Stream first waits for a set time interval called the **Pre-backoff interval** and then begins retrying the request. Time between retries increases based on an exponential backoff algorithm whose base is the **Backoff multiplier**, until the backoff multiplier reaches the **Backoff limit (ms)**. At that point, Cribl Stream continues retrying the request without increasing the time between retries any further.

By default, this Destination has no response codes configured for automatic retries. For each response code you want to add to the list, click **Add Setting** and configure the following settings:

- **HTTP status code**: A response code that indicates a failed request, for example `429 (Too Many Requests)` or `503 (Service Unavailable)`.
- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).

- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Retry timed-out HTTP requests: When you want to automatically retry requests that have timed out, toggle this control on to display the following settings for configuring retry behavior:

- **Pre-backoff interval (ms)**: The amount of time to wait before beginning retries, in milliseconds. Defaults to `1000` (one second).
- **Backoff multiplier**: The base for the exponential backoff algorithm. A value of `2` (the default) means that Cribl Stream will retry after 2 seconds, then 4 seconds, then 8 seconds, and so on.
- **Backoff limit (ms)**: The maximum backoff interval Cribl Stream should apply for its final retry, in milliseconds. Default (and minimum) is `10,000` (10 seconds); maximum is `180,000` (180 seconds, or 3 minutes).

Advanced Settings

Validate server certs: Reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (for example, the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: Compresses the payload body before sending. Defaults to `Yes` (recommended).

Keep alive: By default, Cribl Stream sends `Keep-Alive` headers to the remote server and preserves the connection from the client side up to a maximum of 120 seconds. Toggle this off if you want Cribl Stream to close the connection immediately after sending a request.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests per Worker Process. When Cribl Stream hits this limit, it begins throttling traffic to the downstream service. Defaults to `5`. Minimum: `1`. Maximum: `32`.

Max body size (KB): Maximum size of the request body before compression. Defaults to `4096` KB. You can set this limit to as high as 500 MB (`512000` KB). Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL. The actual request body size might exceed the specified value because the Destination adds bytes when it writes to the downstream receiver. Cribl recommends that you experiment with the **Max body size** value until downstream receivers reliably accept all events.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name-value pairs to pass to all events as additional HTTP headers. Values will be sent encrypted. You can also add headers dynamically on a per-event basis in the `__headers` field. See Internal Fields below.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Custom Format

Choosing `Custom` from the **General Settings** > **Format** drop-down exposes the additional fields described in this section. **Source expression** and most of the rest of the controls define the format of individual events – that is, what data you want to make available from the event. **Batch expression** defines the wrapper object for batched events – that is, how to package events within the request payload. Both expressions **must** be enclosed in backticks.

- **Source expression**: JavaScript expression to evaluate on every event; Cribl Stream will send the result of that evaluation instead of the original event. Sample expression: ``` `${fieldA}, ${fieldB}` ``` (with literal backticks). Defaults to `__httpOut` – that is, the value of the `__httpOut` field. Use the button at right to open a validation modal.
- **Drop when null**: If toggled to `Yes`, Cribl Stream will drop events when the **Source expression** evaluates to `null`.
- **Event delimiter**: Delimiter string to insert between events. Defaults to the newline character (`\n`). Cannot be a space (this will be converted to `\n`).

- **Content type**: Content type to use for requests. Defaults to `application/x-ndjson`. Any content types set in **Advanced Settings > Extra HTTP headers** will override this entry.
- **Batch expression**: Expression specifying how to format the payload for each batch. Defines a wrapper object in which to include the formatted events, such as a JSON document. This enables requests to APIs that require such objects. To reference the events to send, use the `${events}` variable. An example expression to send the batch inside a JSON object would be: `{"items" : [${events}] }`.

Custom Format Example

Suppose that the API you're sending events to requires request payloads in the following form:

```
{
   "Events":[
      {
         "one":"1",
         "two":"2"
      },
      {
         "one":"1.1",
         "two":"2.2"
      }
   ]
}
```

And here's how the events coming into your Webhook Destination look – note that the field names need to change, and `field3` and `field4` need to be dropped:

```
{ "field1": 1, "field2": 2, "field3": 3, "field4": 4 }
{ "field1": 1.1, "field2": 2.2, "field3": 3.3, "field4": 4.4 }
```

To send these events in the required form, you'd do the following:

- Set the **Source expression** to `` `{ "one": "${field1}", "two": "${field2}" }` `` – note that this renames and sends only the desired fields (`one` and `two`).
- Enter the comma (`,`) as your **Event delimiter**.
- Set the **Batch expression** to `` `{ "Events": [${events}] }` ``.

Advanced Format

Choosing `Advanced` from the **General Settings** > **Format** drop-down exposes the JavaScript code blocks described in this section. You can define a format using either one of the code blocks, or both.

- **Format inbound event**: JavaScript code block that receives incoming events (as JavaScript objects) and converts them into strings. Multiple strings are concatenated to form a batch, the batches constituting the payloads of the HTTP requests that the Destination sends to the downstream API.
- **Format outbound payload**: JavaScript code block that receives batches before they are sent out as HTTP request payloads. This gives you the opportunity to modify the payload format from the way it arrives – as a string of concatenated events – to whatever format the downstream API requires.

Code Block Restrictions

Generally speaking, anything forbidden in JavaScript strict mode is forbidden in the code blocks. Specifically, the following are **not allowed**:

- `console`, `eval`, `uneval`, `Function (constructor)`, `Promises`, `setTimeout`, `setInterval`, `global`, `globalThis`, `window`, and `set`.

Code blocks **can** include `for` loops, `while` loops, and JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every`. For further details, see Supported JavaScript Options.

Only skilled JavaScript developers should use the Advanced format. This is to avoid unintended results – such as creating infinite loops, or otherwise failing to return – that could needlessly add to your throughput burden.

Advanced Format Syntax

Here are the syntax conventions for manipulating events and payloads within the code blocks.

Format inbound event Syntax

- `__e`: References the event object.
- `__e['fieldName']`: References an individual field in the event object.
- `__e.asJSON(): string`: Method to safely convert the object to JSON by removing internal content that is not relevant for a downstream system. **Always** use this method instead of directly converting the object to JSON string using `JSON.stringify(__e)`. This avoids circular references that can cause `JSON.stringify(__e)` to fail.

Format outbound event Syntax

In addition to the syntax conventions defined above, the **Format outbound event** code block has its own special conventions:

- `__e['__payload']`: Accesses the `payload` attribute of the JavaScript object that the code block receives. The `__payload` attribute contains the actual batch (payload) to be formatted.
- `__e['__payloadOut']`: After you modify the batch (payload) as desired, assign it to this variable. If the `__payloadOut` attribute is not present, the code block sends the payload as-is. See the JSON Array Example below.

Advanced Format Examples

The **Format inbound event** and **Format outbound payload** code blocks are populated by placeholder JavaScript code. This code expresses the basic flow you must understand to define an Advanced format. Let's examine the code – then you'll be able to see how defining any given Advanced format is just a matter of manipulating events and payloads within the basic flow.

The Format inbound event Code Block

The event to format appears in the first line, referenced as `__e: CriblEvent`. First, you serialize the event as a string; the simplest way to do that is with the method `e.asJSON()`. Then, you assign it back to the object in a variable named `__eventOut`.

```
formatEvent (__e : CriblEvent) {
  // TODO: Serialize the event to a string here.
  let formattedEvent = __e.asJSON();

  // Assign formatted event to __eventOut.
  __e['__eventOut'] = formattedEvent;
}
```

At this point, the `formatEvent()` function ensures that each incoming event will be serialized as a string. (In the more elaborate examples below, the format of that string will be manipulated.) Each new event, in string form, is appended to the last, building a longer string that will become a **batch**. How many events can be in a batch is governed by **Advanced Settings** > **Max events per request**.

Once a batch is complete, the next code block can operate on it.

The Format outbound payload Code Block

Now the `formatPayload ()` function receives an event whose `payload` attribute is the batch produced by the previous code block. You assign that attribute to the `formattedPayload` variable. If you want to manipulate the batch, do that here. Once you have transformed the batch, you assign it to the `__payloadOut` variable, and the Webhook Destination sends the batch out as the payload of an HTTP request.

Transforming batches is not required. To send batches out exactly how they leave the **Format inbound event** code block, just leave the **Format outbound payload** code block blank.

```
formatPayload: (__e : CriblEvent) {
  // Access payload to format here
  let formattedPayload = __e['payload']

  // TODO: Format payload here

  // Assign formatted payload to payloadOut
  __e['payloadOut'] = formattedPayload
}
```

JSON Array Example

Suppose you want to communicate with an API that accepts data in JSON array format.

Use the **Format inbound event** code block to format the events:

```
__e['__eventOut'] = `${__e.asJSON()},` // Serialize to JSON and add trailing comma
```

Use the **Format outbound event** code block to format the payload:

```
// Remove trailing comma and wrap payload in array
__e['__payloadOut'] = `[${payload.substring(0,payload.length-1)}]`
```

Elasticsearch Bulk API Example

The Elasticsearch Bulk API requires each event to be formatted with two lines of output. While this cannot be done with the Webhook Destination's other format options, it is doable with Advanced Format. (The Elasticsearch API Source, Elasticsearch Destination and Elastic Cloud Destination **do** work with the Elasticsearch Bulk API.)

Here's how Bulk API events must be formatted, structurally:

```
{ <action>:  { <metadata> } }
{ <serializedEvent> }
```

Here's an example of an event formatted for Bulk API:

```
{ "index" : { "_index" : "test", "_id" : "1" } }
{ "field1" : "value1", "field2" "value2", "field3" : "value3" }
```

Use the **Format inbound event** code block to format the events:

```
let time=__e['_time'] || Date.now();
__e['_time'] = undefined;
__e['@timestamp'] = C.Time.strftime(time, '%Y-%m-%dT%H:%M:%S.%LZ');
if(typeof __e['host'] === 'string') {
  const host = __e['__host'] || {};
  host.name = __e['host'];
  __e['host'] = host;
}
__e['__eventOut'] = `{"create":{"_index":"${__e['_index'] || 'cribl'}", "pipeline"
```

Leave the **Format outbound event** code block blank.

Splunk HEC Example

Use the **Format inbound event** code block to format the events:

```
    const eventIn = __e;
    const time = eventIn._time;
    const event = eventIn._raw;
    const {host, source, sourcetype, index} = eventIn;
    eventIn._time = undefined;
    eventIn._raw = undefined;
    eventIn.host = undefined;
    eventIn.source = undefined;
    eventIn.sourcetype = undefined;
    eventIn.index = undefined;
    const fields = {};
    const obj2send = {time, index, host, source, sourcetype, event, fields};
    for (const key of ['index', 'host', 'source', 'sourcetype']) {
      // only convert if the key/val is present and is not a string
      const oldValue = obj2send[key];
      if (oldValue != null && typeof oldValue !== 'string') {
        obj2send[key] = JSON.stringify(oldValue);
      }
    }
    for (let key of Object.keys(eventIn)) {
      if(key.startsWith('__'))
        continue;
      let val = eventIn[key];
      if(val === undefined)
        continue;
      if(typeof val === 'object')
        val = JSON.stringify(val);
      obj2send.fields[key] = val;
    }
  __e['__eventOut'] = `${JSON.stringify(obj2send)}\n`;
```

Leave the **Format outbound event** code block blank.

Internal Fields

Cribl Stream 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 Stream will send it to the HTTP endpoint as a metric event. Otherwise, Cribl Stream will send it as a log event.

When **Load balancing** is disabled, if an event that contains the internal field `__url`, that field's value will override the **General Settings** > **Webhook URL** value. This way, you can determine the endpoint URL dynamically.

For example, you could create a Pipeline containing an Eval Function that adds the `__url` field, and connect that Pipeline to your Webhook Destination. Configure the Eval Function to set `__url`'s value to a URL that varies depending on a global variable, or on some property of the event, or on some other dynamically-generated value that meets your needs.

If an event contains the internal field `__headers`, that field's value will be a JSON object containing Name-value pairs, each of which defines a header. By creating Pipelines that set the value of `__headers` according to conditions that you specify, you can add headers dynamically.

For example, you could create a Pipeline containing Eval Functions that add the `__headers` field, and connect that Pipeline to your Webhook Destination. Configure the Eval Functions to set `__headers` values to Name-value pairs that vary depending on some properties of the event, or on dynamically-generated values that meet your needs.

Here's an overview of how to add headers:

- To add "dynamic" (a.k.a. "custom") headers to some events but not others, use the `__headers` field.
- To define headers to add to **all** events, use **Advanced Settings** > **Extra HTTP Headers**.
- An event can include headers added by both methods. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "goat": "Kid A" }`, you'll get both headers.
- Headers added via the `__headers` field take precedence. So if you use `__headers` to add `{ "api-key": "foo" }` and **Extra HTTP Headers** to add `{ "api-key": "bar" }`, you'll get only one header, and that will be `{ "api-key": "foo" }`.

Use Cases

See these examples of configuring a Webhook Destination to integrate with specific services:

- Azure Analytics/Sentinel Integration
- Webhook/BigPanda Integration

- Webhook/Slack Integration

Notes on HTTP-Based Outputs

- To proxy outbound HTTP/S requests, see System Proxy Configuration.
- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After two minutes of the first use, it will throw away the connection and attempt a new one. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), Cribl Stream will establish a new connection for the next request.
- When resolving the Destination's hostname, 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 among destination cluster nodes.

17. Using Integrations

17.1. About Load Balancing

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all endpoints. For example, if FQDNs/hostnames are used as the Destination addresses, and each resolves to 5 (unique) IPs, then each Worker Process will have its # of outbound connections = {# of IPs x # of FQDNs} for purposes of the Destination.

Data is sent by all Worker Processes to a randomly selected endpoint, and the amount sent to each connection depends on these parameters:

1. Respective destination **weight**: If a **Load Weight** is set to `0`, Cribl Stream will not try to connect to that IP (nor to any IPs resolved for that FQDN). When a connection is blocked, Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream will resend the data to a different endpoint. Cribl Stream will block only if **all** endpoints are experiencing problems.

17.2. Destination Backpressure Triggers

This topic describes the conditions that can trigger backpressure in Cribl Stream Destinations. Backpressure exists when a destination receives more data than it can currently send. It typically results from network connectivity problems or when a Destination is receiving data faster than it can forward that data to the downstream service.

When a Destination signals backpressure, it does so to notify the Sources generating data that its buffers are full and that the incoming data flow should be paused until there is room for more data.

You can configure your desired behavior through a Destination's **Backpressure Behavior** drop-down. Where other options are not displayed, Cribl Stream's default behavior is **Block**. For details about all the above behaviors and options, see Persistent Queues.

HTTP-Based Destinations

HTTP-based Destinations trigger backpressure when they encounter errors while connecting or when they receive certain HTTP response codes. If the HTTP call returns a `500` series (`500–599`) HTTP code, the Destination adds the events in the buffer back to its internal buffer and tries to send them again. The retry attempt can also trigger backpressure, depending on the frequency at which this happens.

If the Destinaton receives a `400` series response code, it will drop (not retry) the events, because these response codes indicate an error in payload or formatting (for example). See this Mozilla page on the `400 Bad Request code` for more information.

Finally, HTTP-based Destinations trigger backpressure when their processing loads exceed the specified limit. HTTP-based Destinations handle requests by creating a connection, sending the request payload, and receiving and parsing a response. Because this process can take time, HTTP Destinations allow more than one request to be in progress at a time. When the number of requests in flight exceeds the value set in **Advanced Settings** > **Request concurrency**, the Destination will trigger backpressure.

The following conditions will trigger backpressure:

- Connection failure.
- Connection or request timeouts.
- Data overload – i.e., the Source sends more data than the Destination will accept.
- Reaching the limit on concurrent (in-flight) requests.
- HTTP `500` series responses.

You can set the number of in-flight or concurrent requests in **Advanced Settings** > **Request concurrency**.

HTTP-based Destinations include:

- Splunk HEC
- Elasticsearch
- Wavefront
- SignalFx
- Sumo Logic
- Honeycomb
- Datadog
- NewRelic Ingest Logs & Metrics
- NewRelic Ingest Events
- Azure Sentinel
- CloudWatch Logs
- Webhook
- Grafana Cloud
- Prometheus
- Loki
- Cribl HTTP
- InfluxDB
- Google Cloud Chronicle
- Google Cloud Logging

Filesystem-Based Destinations

Filesystem-based Destinations that send data to cloud-based services trigger backpressure when Cribl Stream can't move a file or upload it to the cloud.

Filesystem-based Destinations queue events in files on disk in a staging directory. When a batch of events is ready for transmission, Cribl Stream closes the file, optionally compresses it, and transmits the file to the downstream service.

For the Filesystem Destination, the batch simply moves from the staging directory to the output directory.

For other filesystem-based Destinations, Cribl Stream uploads the file to a cloud service, such as AWS S3, Azure Blob Storage, or other. For these Destinations, a failure to upload the file will trigger backpressure.

These scenarios include:

- Permissions problem writing to the staging directory – i.e., the user running the Cribl Stream process does not have permission to write to the staging directory.
- The filesystem for the staging or output directory is full (impacts Filesystem Destination only).
- The credentials provided for the cloud service are invalid.
- The Destination doesn't have the permissions required to upload to the cloud service.
- Cloud service permissions issue – i.e., upload fails because the cloud user (identified in credentials) does not have write permissions.
- Network connectivity – The Destination can't communicate with the cloud service due to network connectivity issues.

Filesystem-based Destinations include:

- Filesystem
- Azure Blob Storage
- Google Cloud Storage
- Amazon S3
- MinIO
- CrowdStrike Falcon LogScale
- Amazon Security Lake
- Amazon Data Lakes S3

TCP-Based Destinations

TCP Sources establish a persistent connection with a Destination and send data. These Sources maintain connection to the Destination and will re-establish that connection if they detect a close or write timeout.

If the connection fails, Cribl Stream uses a backoff algorithm to avoid spiking the CPU. This backoff logic makes the first retry at 2 seconds, then gradually increases the time between retries up to a maximum of 60 seconds.

The following conditions can trigger backpressure from a TCP-based Destination:

- Connection failure.
- Authentication failure.
- Write timeout, where a Source sends data but doesn't receive an `ack` at the TCP layer. In this case, Cribl Stream closes the connection to the Destination and tries to re-establish it.
- The Destination receives data from the Source faster than it can send data over the network to the downstream service, causing internal buffers to fill up. This condition can occur due to a slow network

connection or slow processing on the downstream service. When space in the internal buffers becomes available, backpressure is relieved, allowing the Source to send more data.

The following Destinations are TCP-based and adhere to the rules described above. Some Destinations use either TCP or UDP to send messages, but they trigger backpressure and use the backoff algorithm described above only when used with the TCP protocol:

- Splunk Single Instance
- Splunk Load-Balanced
- TCP JSON
- Cribl TCP
- Syslog (TCP or UDP)
- StatsD (TCP or UDP)
- StatsD Extended (TCP or UCP)
- Graphite (TCP or UCP)

Kafka

Kafka is a binary protocol that runs over TCP. The protocol defines three roles – producers, consumers, and brokers.

Producers send data to a Kafka topic (think of it as a message queue). Kafka consumers read data from Kafka topics. Brokers receive data from topic producers and send data to topic consumers – i.e., serve as go-betweens for producers and consumers.

Cribl Stream’s Kafka-based Destinations are producers. They send data to Kafka brokers. Since the protocol is TCP-based, the conditions that trigger backpressure are similar to those of Cribl Stream’s TCP Destinations.

The specifics for connection and request retries are defined by the Kafka JS library in conjunction with the configuration parameters defined for each Destination. In all cases, the Destinations have internal buffers that are sent to the Broker when they’re considered full (on a size and time basis).

The following conditions can trigger backpressure for Kafka-based Sources:

- Failure to connect to the broker, lack of network connectivity, or the broker is down.
- Invalid credentials – Destination connects successfully, but the connection is rejected due to a failure to authenticate.
- Invalid permissions – The credentials assigned do not have proper permissions to send events to the topic.

- The Destination is attempting to send data faster than the broker can receive it. Backpressure will be relieved when internal buffers have room for more events.

Cribl Stream supports the following Kafka-based Destinations:

- Kafka
- Azure Event Hubs
- Confluent Cloud
- MSK

Kafka Retries and PQ

In some situations, Kafka Destinations may take a long time to engage the PQ (1 to 6 minutes by default).

This delay can result from the default settings used with the Kafka Destination. The relevant settings are all in the **Advanced Settings** UI:

- **Connection timeout (ms)**: Defaults to `10000`,
- **Request timeout (ms)**: Defaults to `60000`.
- **Max retries**: Defaults to `5`.

These default settings mean that Cribl Stream will retry five times after a failed connection, with a 10-second timeout between each attempt. It will also retry five files after a failed request, with a timeout of 60 seconds between each attempt.

For faster PQ engagement, update these values. You can use smaller values for **Connection timeout** and **Request timeout** to decrease the interval between retries. You can also use smaller values for **Max retries** or even set it to `0` to eliminate retries altogether.

UDP

UDP is a connectionless protocol, so there is no concept of backpressure at the network layer. As a result, UDP Destinations will never trigger backpressure.

The following Destinations support UDP. Except for SNMP Trap, these Destinations also support TCP, but the UDP versions do not trigger backpressure.

- SNMP Trap
- Syslog
- StatsD
- StatsD Extended

- Graphite

SQS

The SQS (Simple Queue Service) Destination uses the AWS SQS library to send events to the Destination's configured queue. Like Kafka, this Destination acts as a producer that publishes messages to the provisioned SQS queue.

This Destination has five buffers with 10 events per buffer for a total of 50 events that can be queued before backpressure is enabled. The reason that backpressure might be generated for this Destination include:

- Invalid credentials.
- Invalid permissions.
- Queue does not exist and **Create Queue** is disabled.
- Lack of network connectivity.

Amazon Kinesis

Kinesis runs over HTTPS and behaves like Pub/Sub Destinations.

This Destination serializes and publishes events in a newline-delimited JSON format. In the context of the Kinesis Destination, it behaves as a producer where a Kinesis source is considered a consumer. Since the protocol is HTTP-based, the conditions that trigger backpressure are similar to those of other Cribl Stream HTTP Destinations.

The Kinesis library defines the specifics for connection and request retries in conjunction with the configuration parameters defined for each Destination. In all cases, the Destinations have internal buffers that are sent to the Broker when they're considered full (on a size and time basis).

The following conditions can trigger backpressure for Kinesis-based Destinations:

- Failure to connect to the Kinesis service, network connectivity, or the service is down.
- Invalid credentials – Connects successfully but connection is rejected due to a failure to authenticate.
- Invalid permissions – The credentials assigned do not have proper permissions to send events to the Kinesis stream.
- The Destination is attempting to send data faster than the service can receive it. Backpressure will be relieved when internal buffers have room for more events.

Google Cloud Pub/Sub

Google Cloud Pub/Sub is similar to Kafka and Kinesis in that there are producers that send data and consumers that receive data. The Cribl Stream Google Cloud Pub/Sub Destination is a producer that sends data to the Google Cloud Pub/Sub service. Google Cloud Pub/Sub leverages gRCP, however, which uses protocol buffers as both its interface definition language (IDL) and as its underlying message interchange format.

The conditions that trigger backpressure for a Google Cloud Pub/Sub Destination are:

- Failure to connect to the Google Pub/Sub service, network connectivity, or the service is down.
- Invalid credentials – Connects successfully but connection is rejected due to a failure to authenticate.
- Invalid permissions – The credentials assigned do not have proper permissions to send events to the topic.
- The Destination is attempting to send data faster than the service can receive it. Backpressure will be relieved when internal buffers have room for more events.

OpenTelemetry

OpenTelemetry supports gRPC or HTTP to send messages. For the gRPC protocol, the same rules described for other TCP Destinations generally apply to OpenTelemetry. The key difference is that this Destination uses the OpenTelemetry library, which handles write timeouts and reconnection logic.

For the HTTP protocol, the same rules apply to our HTTP Destinations for retries based on connectivity or `500` series response codes.

Load-Balanced Destinations

HTTP-Based Load-Balanced Destinations

Some HTTP-based Destinations support two modes – single host and multiple load-balanced hosts.

With a load-balanced version of an HTTP Destination, Cribl Stream spreads events across all configured Destinations, using a weighted algorithm. Under normal circumstances, this algorithm spreads the load across the configured Destinations according to the weight assigned to the Destination.

When one of the load-balanced Destinations is out of service (due to a write timeout or `500` series response code - buffers full), Cribl Stream removes that Destination from the list of hosts and moves any in-flight buffers to the next available host. Cribl Stream will try to resend events to the downed host in the background, and it will resume sending data to that host once the connection is restored.

HTTP Destinations that support load balancing include:

- Splunk HEC
- Elasticsearch
- Cribl HTTP

TCP Load-Balanced Destinations

Some TCP-based Destinations support two modes – single host and multiple load-balanced hosts.

With a load-balanced TCP Destination, Cribl Stream spreads events across all configured Destinations using a weighted algorithm. Under normal circumstances, this algorithm spreads the load across the configured Destinations according to the weight assigned to each Destination.

When one of the load-balanced Destinations is out of service (due to a write timeout or connection failure), Cribl Stream removes that Destination from the list of hosts and moves any in-flight buffers to the next available host. Cribl Stream will try to reconnect to the downed host and will resume sending data to that host once the connection is restored.

TCP load-balanced Destinations can be configured with one or multiple receivers. If one or more receivers go down, Cribl Stream will continue sending data to any healthy receivers.

TCP load-balanced Destinations include:

- Splunk Load-Balanced
- TCP JSON
- Cribl TCP
- Syslog (TCP version only)

Output Router

The Output Router is a wrapper destination that groups together 1..*N* Destinations. It's used in conjunction with event filters to route data to a selected Destination. The Output Router Destination emits backpressure only when one of its downstream Destinations is blocking.

17.3. Persistent Queues

Cribl Stream's persistent queuing (PQ) feature helps minimize data loss if a downstream receiver or is unreachable or slow to respond, or (when configured on Sources) during backpressure from Cribl Stream internal processing. PQ provides durability by writing data to disk for the duration of the outage, and then forwarding it upon recovery.

Persistent queues can be enabled:

- On Push Sources.
- On Streaming Destinations. (Sources can also take advantage of a Destination's queue to keep data flowing.) For details, see Persistent Queues Support.

As a paid feature, persistent queues are available:

- On customer-managed (on-prem) deployments with at least a Standard license.
- On Cribl.Cloud with an Enterprise plan.

Persistent Queues Supplement In-Memory Queues

Persistent queues trigger differently on the Destination versus Source side.

Destination Side

On Cribl Stream Destinations, in-memory buffers help absorb temporary imbalances between inbound and outbound data rates. For example, if there is an inbound burst of data, a Worker Process will store events in Destination buffers, and will then output them at the rate that the downstream receiver can catch up.

Only when buffers are saturated will the Destination impose backpressure upstream. (This threshold varies per Destination type.) This is where enabling persistent queues helps safeguard your data.

Destinations Without PQ

You can configure each Destination's backpressure behavior to one of **Block**, or **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 Stream 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 or 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 buffer is full, the Source or Destination will write its data to disk. Then, when the downstream receiver or Cribl process is ready, the Source/Destination will start draining its queue.

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

Source Side

Push Sources' config modals also provide a PQ option to supplement in-memory buffering. When you enable PQ, you can choose between two trigger conditions:

- `Always On` mode will use PQ as a disk buffer for **all** events.
- `Smart` mode will engage PQ only upon backpressure from downstream receivers or Cribl internal processes.

> To learn more about PQ options on both the Source and Destination sides, see Planning for Persistent Queues. For implementation details, see Configuring Persistent Queues.

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), Azure Data Explorer (when you select Batching mode), and SNMP Trap.

For specifics on both sides, see Persistent Queues Support.

- 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.
- 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).

Beyond the failure scenarios above: If you turn off PQ on a Source or Destination (or shut down the whole Source/Destination) before the disk queue has drained, the queued events will be orphaned on disk.

17.3.1. Planning For Persistent Queues

The Insider's Guide

This page delves into:

- Use cases for implementing persistent queues (PQ).
- Details about PQ behavior on Sources and Destinations, with various configuration and tuning options.
- Choices you should make before enabling PQ.
- Implications of those choices for your system's requirements and performance.

For granular configuration procedures, see Configuring Persistent Queueing.

Source-Side PQ

Source-Side PQ can serve as a stopgap for upstream senders that don't provide their own buffering, or whose small buffers cannot queue for very long. This applies to several syslog senders and HTTP-based Sources.

Here, PQ prevents data loss when the Source is unable to handle backpressure for an extended period. Backpressure might be triggered by error conditions, or by degraded performance (slowdowns or short backups), in downstream Cribl resources or outside receivers.

Always On Versus Smart Mode

Where senders do not provide their own buffering, and where Destination-Side PQ is not enabled: It is a good practice to enable Push Sources' PQ option, in either `Smart` or `Always On` mode. But which mode should you choose?

`Always On` mode is a good match for highly valuable data (such as security data), and for data ingested via UDP. (Relevant UDP Sources are Raw UDP, SNMP Trap, Metrics, and Syslog with UDP enabled.)

`Smart` mode might be sufficient for lower-grade data, such as events that are overwhelmingly `debug`-level.

Always On Requirements

`Always On` mode provides the most reliable data delivery, but you should plan for its impacts on throughput, memory, and hosts' disk resources.

Throughput

Each Source configured in `Always On` mode imposes a significant performance penalty on Workers' data ingestion rate, as compared to Sources without PQ enabled. This primarily involves the overhead to write to the filesystem. As a starting point, assume that each event will be burdened with approximately 1 second of added latency.

Memory

`Always On` mode can also increase memory demands, as compared to Sources without PQ enabled. If you find memory to be a constraint, you can experiment with tuning the **Max buffer size** in each Source's UI, or the `maxBufferSize` configuration key. However, smaller buffers will reduce the efficiency of disk writes, adding more latency.

Disk

`Always On` also imposes more disk requirements than `Smart` mode. Cribl generally recommends that you allocate each Worker Node enough disk space for at least 1 day's worth of queued throughput. For example: If one Source is sending 1 TB/day to two Worker Processes (evenly weighted), you should provision at least 500 GB queue storage for each Worker Process. Expand this basic calculation for additional Sources, and expand or contract it based on the typical length of outages you experience.

Cribl also recommends that you deploy the highest-performance storage available. This prevents IOPS (input/output operations per second) and throughput bottlenecks for both read and write operations.

Compression

Enabling the PQ compression option conserves disk space, with a compression ratio of about 8:1. But you get these savings at the cost of worsening the performance hit. Navigate this trade-off based on which factor is most valuable in your deployment.

Smart Mode Details

`Smart` mode engages only when a Source receives an unhealthy signal from a downstream process (Pipeline, Destination down or blocked, etc.). When engaged, `Smart` mode's latency penalty is about the same as with `Always On` mode. But this latency is highly variable, because `Smart` mode engages intermittently. So expect less latency overall, with less predictability – spikes and valleys.

`Smart` mode's CPU performance penalty is about twice that of `Always On`, because of the overhead to switch queueing on and off. But again, this is highly variable, because performance will degrade only when

PQ engages/disengages. As with `Always On`, if memory is a constraint, you can experiment with reducing **Max buffer size** (UI) or `maxBufferSize` (config-file) values. Compression is supported in `Smart` mode, as in `Always On` modes.

Smart Mode Interaction with Multiple Destinations

If you enable `Smart` mode on a Source that's routed to multiple Destinations, you should plan around the unintended interaction that PQ can cause among these Destinations:

If at least one connected Destination sends an unhealthy signal, this Source will queue new events to disk even if some peer Destinations remain up. This will delay data delivery to the peer healthy Destinations. The same interaction can occur between Destinations grouped in a connected Output Router.

To prevent this unintended behavior, you can either:

- Disable PQ on the Source, and instead configure PQ on individual Destinations; or
- Configure separate Worker Groups to connect this Source to separate Destinations.

How Source PQ Works

Once enabled on a specific Source, in `Smart` mode, PQ will engage when any of the following occur:

- A connected Destination is fully down, and Destination-side PQ is not enabled.
- A connected Destination is slow, or blocking.
- A downstream Cribl Pipeline/Route is slow.

These triggers are not relevant if you've enabled Source PQ in `Always On` mode. In that case – it's always on.

When PQ engages on a Source:

- Events queue **before** any pre-processing Pipeline takes effect.
- Metrics (data in) will show `0` bytes/events while events are buffered to the disk queue.
- Cribl Stream Pipelines/Routes will process the data with a delay, when PQ later drains.

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

If you turn off Source PQ (or the whole Source) before its queues drain, any events still queued on disk will be orphaned.

Destination-Side PQ

You can enable PQ on supported Destinations to safeguard data during extended errors, and to smooth out CPU load spikes when a Destination recovers from a transient outage.

Unlike Source-side PQ, Destination-side PQ will engage only when the Destination is **completely** unavailable, not just slow.

Destination PQ is a good match for receivers like Splunk, where there are known outages. Because it has no `Always On` mode, Destination PQ imposes less overall performance overhead than Source PQ.

As with Source PQ, size your disk capacity according to your risk tolerance, but Cribl's initial recommendation is to size for at least 1 day's worth of storage on each Worker Node.

Fallback (Queue-Full) Behavior

Unlike Source PQ, on Destinations where you've enabled PQ, you also select a fallback **Queue-full behavior**.

With the default `Block` option, if a Destination's persistent queue fills up, the Destination will block back to the corresponding Source. (Remember that a Destination can be fed by multiple Sources.) If the Source has its own PQ option enabled, it will begin queueing data. If not, the block signal will go all the way back to connected senders.

If you instead set the **Queue-full behavior** to `Drop new data`, Cribl Stream will simply discard data intended for the unavailable Destination, but will still allow the data to proceed to parallel Destinations.

Good use cases for `Block`: A simple Cribl Stream Route relaying Universal Forwarder data to a Splunk receiver; or, any Destination where you absolutely **must not** lose data.

How Destination PQ Works

If a Destination does not have load balancing enabled, PQ will engage upon a blocking or unavailable signal from the downstream receiver.

But with a load-balanced Destination, PQ will engage only if **all** of the Destination's downstream receivers are down or blocked. If some individual targets in the LB Destination are up, Cribl Stream will attempt to discover those receivers and deliver data to them.

When PQ is enabled on a specific Destination:

- PQ will engage when the Destination is completely unavailable, but not when it is simply slow.

- Files are written to disk **after** any post-processing Pipeline takes effect. (This is the mirror image of Source PQ, which writes to disk **before** pre-processing Pipelines.)
- New incoming data still flows to **other** Destinations that are up.
- When the Destination recovers, the default draining behavior is FIFO (first in, first out) But you can toggle off this **Strict ordering** default– and doing so enables a configurable **Drain rate limit (EPS)**.
- A **Clear Persistent Queue** button is available on each Destination, in case you need to clear out the queue's contents. This is destructive – it recovers the disk space, but abandons the queued data. Treat it as a panic button.

Destination PQ Better Practices

- Cribl generally recommends enabling the PQ **Compression** option on Destinations. (Disable compression if you find that it adds excessive latency.)
- When draining large volumes after recovery from an outage, you might find that PQ is crushing its Destination. In this case, disable **Strict ordering**, and tune the **Drain rate limit (EPS)** to maintain reasonable throughput for new data.

Better Practices for Source and/or Destination PQ

If your Source(s) and Destination(s) both support PQ, which side should you enable?

Enabling Source-side PQ (in `Always On` mode) is the most versatile and reliable choice. Source PQ is further upstream than Destination PQ, and can safeguard data bound for multiple Destinations.

However, you might opt for Destination-side PQ if your senders provide their own buffering, or if you don't want to add the system resources that Source Always On mode requires.

Enabling **both** Source- and Destination-side PQ on a Route is likely overkill. It can also complicate diagnosing issues.

Warning for Auto-Scaling Environments

With **persistent** storage, events on disk survive a Cribl Stream/Edge or OS restart. But you cannot count on this with ephemeral/auto-scaling environments. Stream Workers must **not** be destroyed while data has not drained from a PQ.

PQ Status Notifications and Monitoring

With an Enterprise or Standard license, you can configure Notifications that will be sent when PQ files exceed a configurable percentage of allocated storage, or when a Destination reaches the queue-full state described above.

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

Destination PQ Notifications

With Destination PQ enabled, you can configure Notifications around two trigger conditions. Cribl recommends that you create both:

- Destination Backpressure Activated. This will send Notifications whenever backpressure engages, causing blocked or dropped events. It will also send Notifications when a queue fills up, falling back to blocked or dropped events.
- Persistent Queue Usage (saturation). Note that you can set these Notifications for **multiple** saturation levels.

Source PQ Alerting

Currently, Cribl Stream does not support Notifications around Source PQ states. However, if you've enabled `Always On` mode, alerts that PQ has engaged would be pointless – because all of the Source's incoming data is automatically written to the disk queue.

Also, if desired: On Fleets or customer-managed (on-prem) Worker Groups, you can enable the internal CriblLogs Source, create a Route to forward Cribl's internal logs to a downstream service of your choice, and configure alerts from there.

Monitoring PQ

Cribl Stream's native Monitoring dashboards interact with PQ in particular ways. When PQ engages, assume that your sole sources of truth about Sources' or Destinations' throughput are the dashboards at:

- **Monitoring** > **System** > **Queues (Sources),** and
- **Monitoring** > **System** > **Queues (Destinations)**.

Other Monitoring resources become unreliable during queueing. These include the top-level **Monitoring** dashboard, and:

- **Monitoring** > **Data** > **Sources,** or
- **Monitoring** > **Data** > **Destinations**.

These will not accurately represent queued data because, once PQ engages, data will be queued to disk, rather than arriving into Pipelines and downstream resources (with Source PQ) or into the affected Destinations (with Destination PQ). While the data is queueing, the two above dashboards for:

- Affected Sources will show no events per second (EPS) or bytes ingested while their data is queueing.
- Affected Destinations will show EPS and bytes sent while their data is queueing, but these metrics can be misleading. Here, they measure data going into the queue, but not proceeding to downstream receivers.

Some other Monitoring quirks to keep in mind:

- Activity in the **Monitoring** > **System** > **Queues (Sources)** and **Monitoring** > **System** > **Queues (Destinations)** dashboards sometimes shows up only when queues drain.
- Cribl Stream currently does not provide a Monitoring dashboard for disk queues' own volumes.

These gaps provide another reason to enable PQ Notifications where available.

Remember that with **or** without PQ enabled, Monitoring dashboards do not show usage of integrations' in-memory buffers.

After the unhealthy condition resolves, and disk queues start to drain, dashboards will resume accurately representing delivery rates and volumes.

Metrics and Capture Limitations

Finally, while PQ provides many benefits, also note the following limitations:

- While events go to a Source-side queue, Cribl Stream does not record metrics about that Source's inbound data (events/second or bytes).
- While events go to a Destination-side queue, metrics reflect events/second and bytes going into that Destination's queue, not proceeding beyond Cribl Stream.
- Data captures might not work while a Destination is blocked.

17.3.2. Configuring Persistent Queues

The following sections outline configuring PQ for Cribl.Cloud Workers managed by Cribl, and then for on-prem and hybrid Cribl.Cloud Workers.

Configuring PQ for Cribl-Managed Cribl.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 on each Source and Destination that supports it.

Sources

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

On Cribl-managed Workers, this tab exposes only the **Enable Persistent Queue** toggle. When enabled, PQ is automatically configured in `Always On` mode, with a maximum queue size of 1 GB of data per PQ-enabled Source, per Worker Process. The 1 GB limit is on uncompressed inbound data, and no compression is applied to the queue.

These options are not configurable. If you want to finely configure the maximum queue size, compression, PQ mode, and other options, use a hybrid Group.

Destinations

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

On Cribl-managed Workers, the resulting **Persistent Queue Settings** tab exposes only a **Clear Persistent Queue** button. Once enabled, PQ is automatically configured with a maximum queue size of 1 GB of uncompressed data per PQ-enabled Destination, per Worker Process. The 1 GB limit is on uncompressed outbound data, and no compression is applied to the queue. If the queue fills up, Cribl Stream will block data flow.

These options are not configurable. If you want to finely configure the maximum queue size, compression, queue-full fallback behavior, and other options, use a hybrid Group.

If you want to disable persistent queuing:

1. Wait for the queue to fully drain.
   (The **Clear Persistent Queue** button is available as a destructive fallback. This will free up disk space, but will do so by deleting the queued data **without** sending it on to receivers.)
2. In the config modal's **General Settings**, select a different **Backpressure behavior**.
3. Re-save the config.

Logs

Cribl internal logs for Cribl-managed Workers might include entries labeled `Revived PQ Worker Process`. These indicate where a Cribl-managed Worker was newly assigned to drain a persistent queue left over from a shut-down Worker Node. You can use these log events to troubleshoot Cloud PQ issues.

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. (Cribl-managed Cloud Workers have a simpler configuration.) 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 **Enable Persistent Queue** to 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 How Does Persistent Queuing Work Inside Cribl Stream? blog post.

Source-Side PQ Only

Mode: Select a condition for engaging persistent queues.

- `Always On`: This default option will always write events to the persistent queue, before forwarding them to Cribl Stream's data processing engine.
- `Smart`: This option will engage PQ only when the Source detects backpressure from Cribl Stream's data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the sender and writing the queue to disk. Defaults to `1000`. (This buffer is per connection, not just per Worker Process – and this can dramatically expand memory usage.)

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

Always On versus Smart Mode

In Cribl Stream 4.1 and later, Source-side PQ defaults to `Always on` mode when you configure a new Source. `Always on` offers the highest data durability, by minimizing backpressure and potential data loss on senders. However, this mode can slightly slow down data throughput, and can require you to provision more machines and/or faster disks.

If you are upgrading from a pre-4.1 version where `Smart` mode was the default, this change does not affect your existing Sources' configurations. However:

- If you create new Stream Sources programmatically, and you want to enforce the previous `Smart` mode, you'll need to update your existing code.
- You can optimize Workers' startup connections and CPU load at **Group Settings** > Worker Processes.

Consider switching to `Always on` mode if you find `Smart` mode causing Source-side PQ to frequently engage, and you've configured PQ with a large **Max file size** (e.g., 1 GB). However, a second option here is to retain `Smart` mode, but reduce the **Max file size** to a value no higher than the default `1 MB`.

(Source-side PQ excessively triggers backpressure only when it reaches the last queued file **and** that file is large. Keeping this threshold low prevents that condition.)

Common PQ Settings (Source and Destination Sides)

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl Stream applies the default `1 MB`.

Max queue size: The maximum amount of disk space that the queue is allowed to consume, on each Worker Process. Once this limit is reached, this Source or Destination will stop queueing data; Sources will block, and Destinations will apply your configured **Queue-full behavior**. Defaults to `5 GB`. Enter a numeral with units of KB, MB, GB, or TB. For details, see Optimizing Max Queue Size.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this value, Cribl 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.

Optimizing Max Queue Size

When you upgrade Fleets, and customer-managed (on-prem) and hybrid Groups, to Cribl Stream 4.4 and later: Any undefined PQ **Max queue size** values in pre-4.4 Source/Destination configs will be set to the new 5 GB default value. This is a guardrail to control queues' consumption of host machines' disk space.

Adjust this default to match your needs. If some of your integrations' persistent queues had an undefined **Max queue size**, queues larger than 5 GB will trigger PQ fallback behavior until you resize this limit.

Also in Cribl Stream 4.4 and later, this field's highest accepted value is defined in the new **Group Settings** > **General Settings** > **Limits** > **Storage** > **Max PQ size per Worker Process** field. The corresponding `limits.yml` key is `maxPQSize`. Consult Cribl Support before increasing that limit's 1 TB default.

The changes here do not affect any existing config with a defined **Max queue size**, and do not affect Cribl-managed Cloud Workers' nonconfigurable 1 GB **Max queue size**.

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 Stream 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 "panic" button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear - because this will free up disk space by permanently deleting the queued data, without delivering it to downstream receivers. (Appears only after **Output ID** has been defined.)

This section's remaining controls are available in Cribl Stream 4.1 and later.

Strict ordering: The default `Yes` position enables FIFO (first in, first out) event forwarding. When receivers recover, Cribl Stream 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 **Group Settings** > Worker Processes.

Minimum Free Disk Space

For queuing to operate properly, you must provide sufficient disk space. In distributed mode, you configure the minimum disk space to support queues (and other features) on each Worker Group, at **Group Settings** > **General Settings** > **Limits** > **Min free disk space**. If available disk space falls below this threshold, Cribl Stream will stop maintaining persistent queues, and data loss will begin. The default minimum is 5 GB.

17.3.3. Persistent Queues Support

This page identifies Cribl integrations that do, and don't, support persistent queues.

Persistent Queues Support by Source Type

The dividing line here is simple:

- Cribl Push Sources, which ingest streaming data, support PQ.
- Pull and Collector Sources, which ingest data intermittently and/or from static files, omit PQ support.

Persistent Queues 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 Stream will simply drop corresponding events. Cribl Stream 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:

- Amazon Cloudwatch Logs
- Amazon S3
- Amazon SQS
- Azure Blob Storage
- Azure Monitor Logs
- Cribl HTTP
- CrowdStrike Falcon LogScale

- Datadog
- Data Lake > S3
- Elasticsearch
- Google Chronicle
- Google Pub/Sub
- Grafana Cloud
- Honeycomb
- InfluxDB
- Loki (Logs)
- New Relic Ingest: Logs & Metrics
- New Relic Ingest: Events
- OpenTelemetry
- Prometheus
- SentinelOne DataSet
- SignalFx
- Splunk HEC
- Sumo Logic
- WaveFront
- Webhook

TCP Load-Balanced Destinations

TCP load-balanced Destinations can be configured with one or multiple receivers. If one or more receivers go down, Cribl Stream 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 Stream 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 that use the UDP protocol, do not support PQ. These include:

- Amazon S3 Compatible Stores
- Data Lakes > Amazon S3
- Data Lakes > Amazon Security Lake
- Azure Blob Storage
- Azure Data Explorer when in batching mode
- Filesystem/NFS
- Google Cloud Storage
- MinIO
- SNMP Trap
- StatsD (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 Stream gets no indication whether the receiver received an event.

Other Destinations

Other Destinations that write to a single receiver (without load balancing enabled) generally engage PQ based on these trigger conditions:

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

17.4. 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:

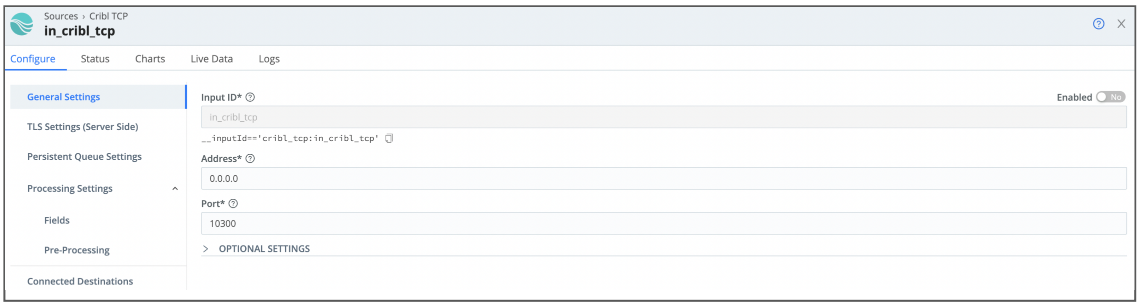
- Configure Cribl TCP Source on Cribl Stream to receive data from the Edge Node.
- Configure the Exec Source on Cribl Edge to collect data on the Edge Node.
- Configure the Cribl TCP Destination on Cribl Edge to send data to Cribl Stream.
- Configure a Route to Send the Data.

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.)



Input ID*: in_cribl_tcp

Enabled: No

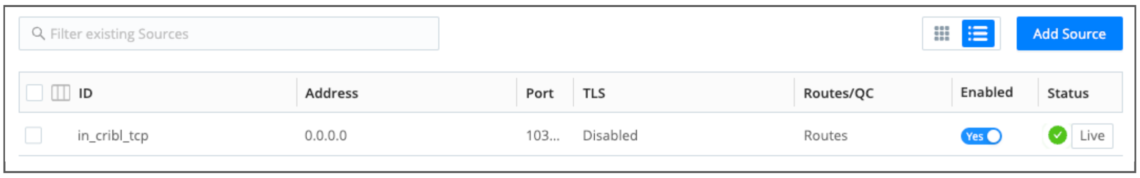
`__inputId=='cribl_tcp:in_cribl_tcp'`

Address*: 0.0.0.0

Port*: 10300

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.



ID	Address	Port	TLS	Routes/QC	Enabled	Status
in_cribl_tcp	0.0.0.0	103...	Disabled	Routes	Yes	Live

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

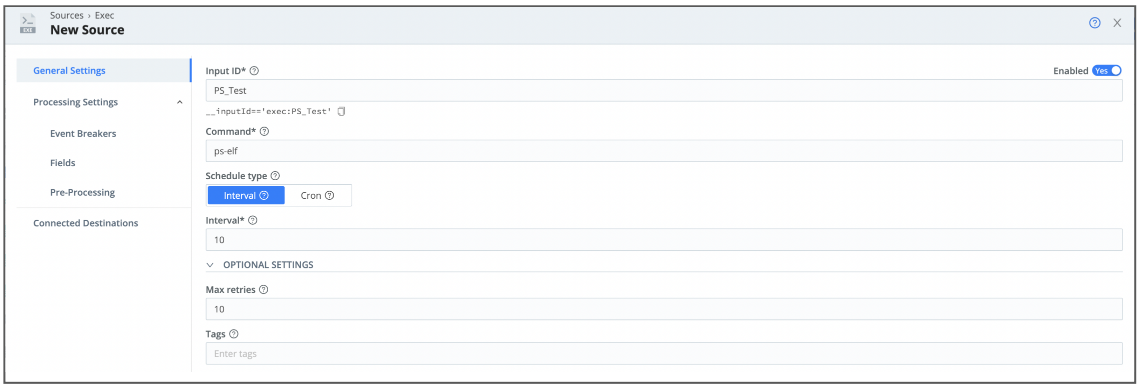
Next, we'll configure the Exec Source on your Edge Node. This Source will break the incoming streams of data into discrete events, and send them to Cribl Stream.

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.



Input ID*: PS_Test

`__inputId=='exec:PS_Test'`

Enabled: Yes

Command*: ps-elf

Schedule type: [✓ Interval] Cron

Interval*: 10

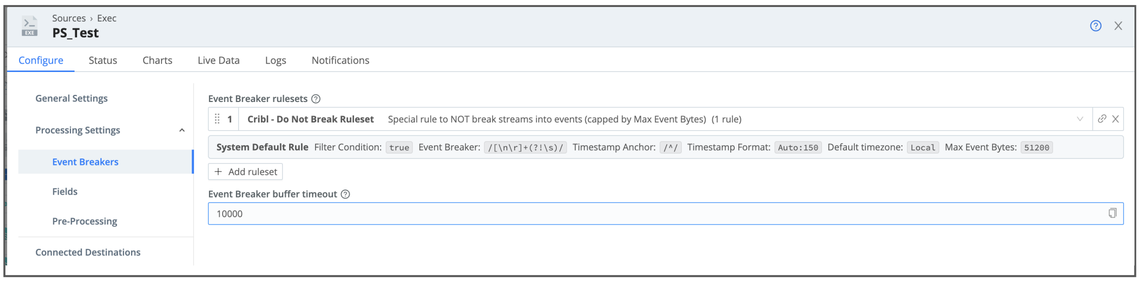
Max retries: 10

Tags: Enter tags

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`.



Sources › Exec
PS_Test

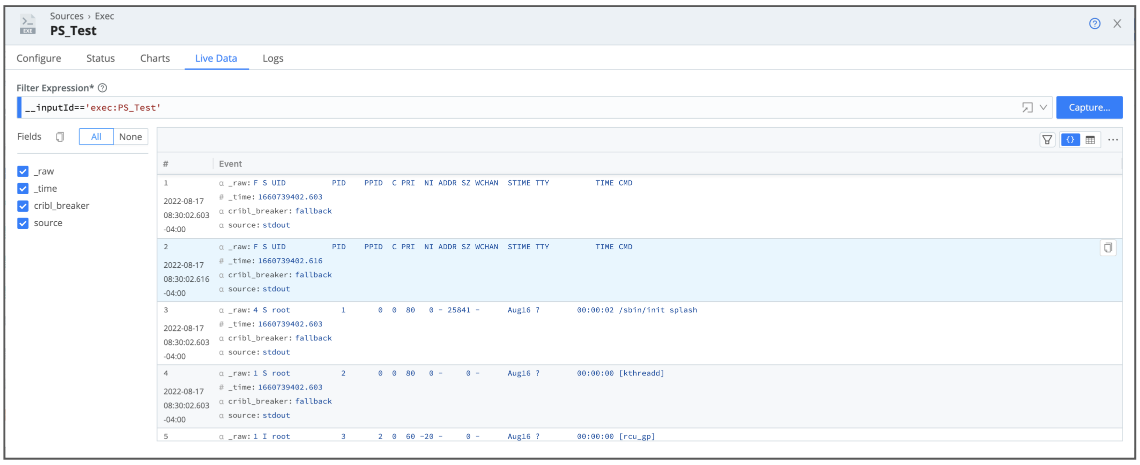
Configure | Status | Charts | Live Data | Logs | Notifications

General Settings
Processing Settings
Event Breakers
Fields
Pre-Processing
Connected Destinations

Event Breaker rulesets
1 **Cribl - Do Not Break Ruleset** Special rule to NOT break streams into events (capped by Max Event Bytes) (1 rule)
System Default Rule Filter Condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp Anchor: `/^/` Timestamp Format: `Auto:150` Default timezone: `Local` Max Event Bytes: `51200`
+ Add ruleset
Event Breaker buffer timeout
10000

Apply an Event Breaker

Next, preview your data on the modal’s **Live Data** tab.



Sources › Exec
PS_Test

Configure | Status | Charts | Live Data | Logs

Filter Expression*
`__inputId=='exec:PS_Test'`
Capture...

Fields: All | None
☑ _raw
☑ _time
☑ cribl_breaker
☑ source

```
#  Event
1  _raw: F S UID          PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
2022-08-17
08:30:02.603
-04:00
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
2  _raw: F S UID          PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
2022-08-17
08:30:02.616
-04:00
   _time: 1660739402.616
   cribl_breaker: fallback
   source: stdout
3  _raw: 4 S root           1       0  0  80   0 - 25841 -      Aug16 ?        00:00:02 /sbin/init splash
2022-08-17
08:30:02.603
-04:00
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
4  _raw: 1 S root           2       0  0  80   0 -     0 -      Aug16 ?        00:00:00 [kthreadd]
2022-08-17
08:30:02.603
-04:00
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
5  _raw: 1 I root           3       2  0  60 -20 -     0 -      Aug16 ?        00:00:00 [rcu_gp]
```

Preview Live Data

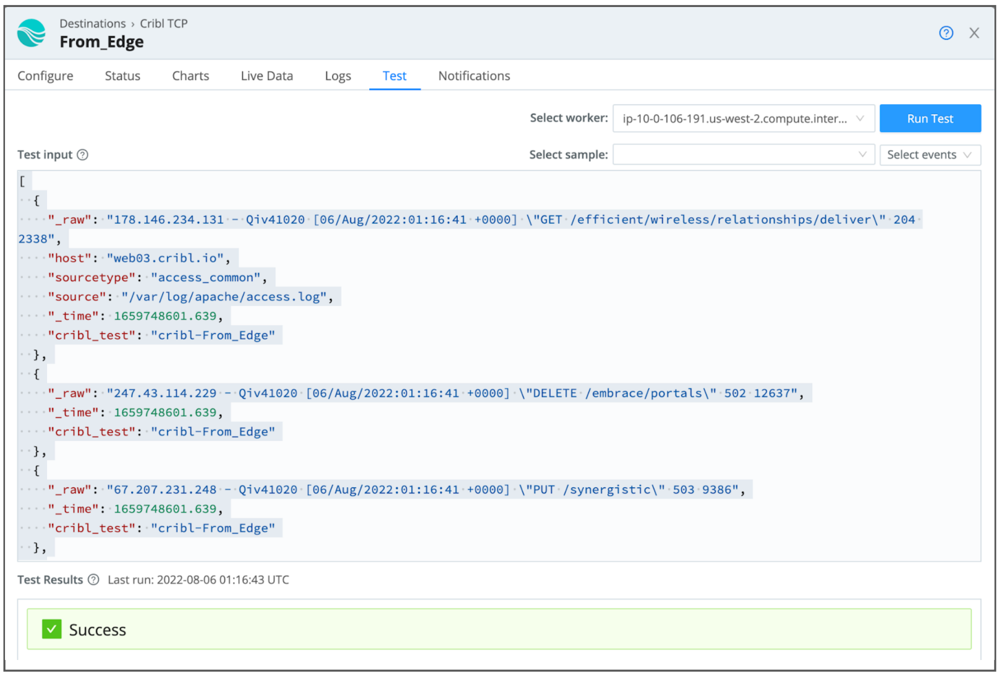
Configure the Cribl TCP Destination on Cribl Edge

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.



Select worker: ip-10-0-106-191.us-west-2.compute.inter...

Select sample:

```
[
  {
    "_raw": "178.146.234.131 - Qiv41020 [06/Aug/2022:01:16:41 +0000] \"GET /efficient/wireless/relationships/deliver\" 204 
2338",
    "host": "web03.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
    "_time": 1659748601.639,
    "cribl_test": "cribl-From_Edge"
  },
  {
    "_raw": "247.43.114.229 - Qiv41020 [06/Aug/2022:01:16:41 +0000] \"DELETE /embrace/portals\" 502 12637",
    "_time": 1659748601.639,
    "cribl_test": "cribl-From_Edge"
  },
  {
    "_raw": "67.207.231.248 - Qiv41020 [06/Aug/2022:01:16:41 +0000] \"PUT /synergistic\" 503 9386",
    "_time": 1659748601.639,
    "cribl_test": "cribl-From_Edge"
  },
```

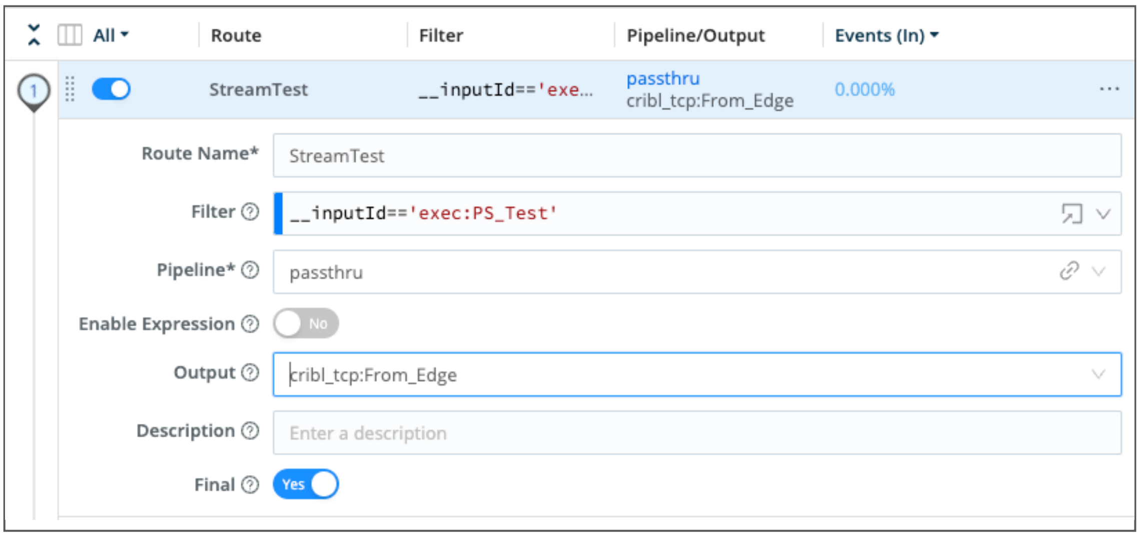
Last run: 2022-08-06 01:16:43 UTC

✓ Success

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.



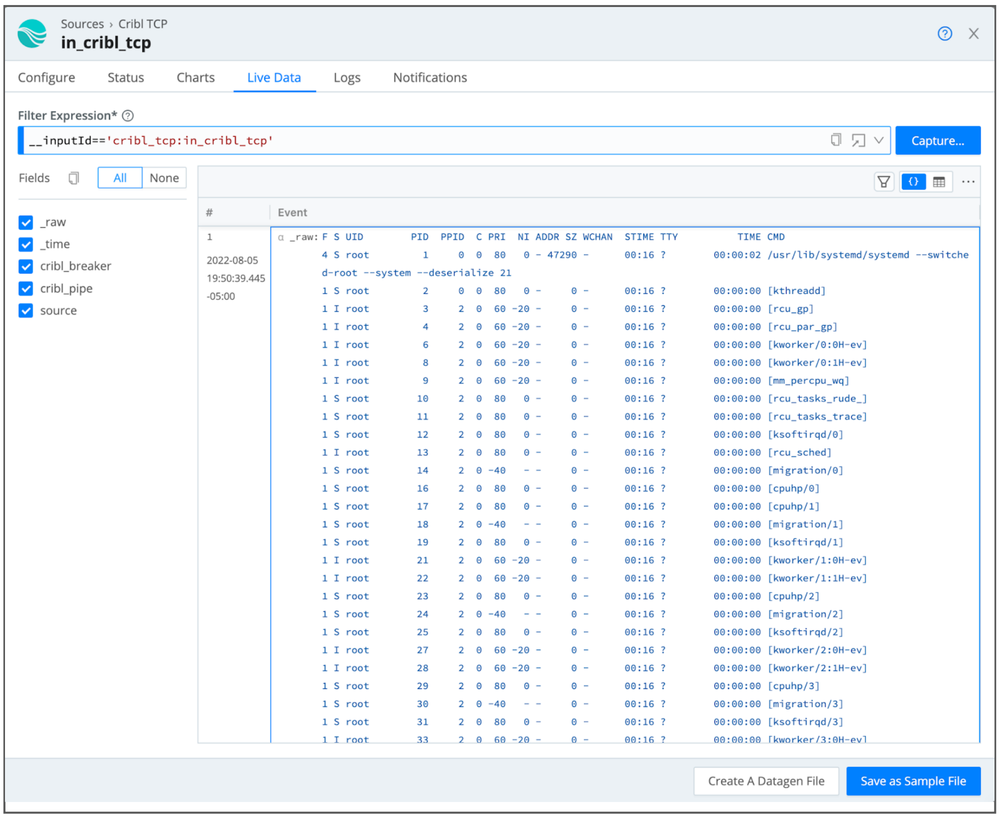
Route	Filter	Pipeline/Output	Events (In)
StreamTest	`__inputId=='exe...`	passthru cribl_tcp:From_Edge	0.000%

Route Name*	StreamTest
Filter	`__inputId=='exec:PS_Test'`
Pipeline*	passthru
Enable Expression	No
Output	cribl_tcp:From_Edge
Description	Enter a description
Final	Yes

Routing your data

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.

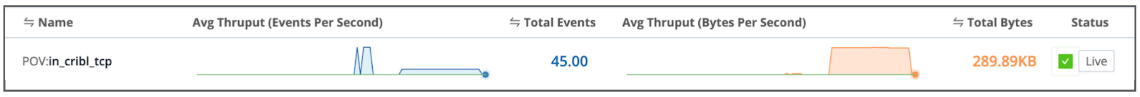


```
__inputId=='cribl_tcp:in_cribl_tcp'
```

```
1
2022-08-05
19:50:39.445
-05:00
α _raw: F S UID        PID  PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
        4 S root         1     0  0  80   0 - 47290 -      00:16 ?        00:00:02 /usr/lib/systemd/systemd --switche
        d-root --system --deserialize 21
        1 S root         2     0  0  80   0 -     0 -      00:16 ?        00:00:00 [kthreadd]
        1 I root         3     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [rcu_gp]
        1 I root         4     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [rcu_par_gp]
        1 I root         6     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/0:0H-ev]
        1 I root         8     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/0:1H-ev]
        1 I root         9     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [mm_percpu_wq]
        1 S root        10     2  0  80   0 -     0 -      00:16 ?        00:00:00 [rcu_tasks_rude_]
        1 S root        11     2  0  80   0 -     0 -      00:16 ?        00:00:00 [rcu_tasks_trace]
        1 S root        12     2  0  80   0 -     0 -      00:16 ?        00:00:00 [ksoftirqd/0]
        1 I root        13     2  0  80   0 -     0 -      00:16 ?        00:00:00 [rcu_sched]
        1 S root        14     2  0 -40   - -     0 -      00:16 ?        00:00:00 [migration/0]
        1 S root        16     2  0  80   0 -     0 -      00:16 ?        00:00:00 [cpuhp/0]
        1 S root        17     2  0  80   0 -     0 -      00:16 ?        00:00:00 [cpuhp/1]
        1 S root        18     2  0 -40   - -     0 -      00:16 ?        00:00:00 [migration/1]
        1 S root        19     2  0  80   0 -     0 -      00:16 ?        00:00:00 [ksoftirqd/1]
        1 I root        21     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/1:0H-ev]
        1 I root        22     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/1:1H-ev]
        1 S root        23     2  0  80   0 -     0 -      00:16 ?        00:00:00 [cpuhp/2]
        1 S root        24     2  0 -40   - -     0 -      00:16 ?        00:00:00 [migration/2]
        1 S root        25     2  0  80   0 -     0 -      00:16 ?        00:00:00 [ksoftirqd/2]
        1 I root        27     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/2:0H-ev]
        1 I root        28     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/2:1H-ev]
        1 S root        29     2  0  80   0 -     0 -      00:16 ?        00:00:00 [cpuhp/3]
        1 S root        30     2  0 -40   - -     0 -      00:16 ?        00:00:00 [migration/3]
        1 S root        31     2  0  80   0 -     0 -      00:16 ?        00:00:00 [ksoftirqd/3]
        1 I root        33     2  0  60 -20 -     0 -      00:16 ?        00:00:00 [kworker/3:0H-ev]
```

Data flow in the Source

You can also check the Monitoring page's **Data** submenu, to isolate the throughput on your Source.



Name	Avg Thruput (Events Per Second)	Total Events	Avg Thruput (Bytes Per Second)	Total Bytes	Status
POV:in_cribl_tcp		45.00		289.89KB	✓ Live

Monitoring the Source throughput

17.5. Using Collectors

17.5.1. Using S3 Storage And Replay

Cribl Stream's Replay options offer organizations fundamentally new ways to manage data, by providing an easy way to selectively ingest, **and re-ingest**, data into systems of analysis. Let's walk through how to use this feature, step by step.

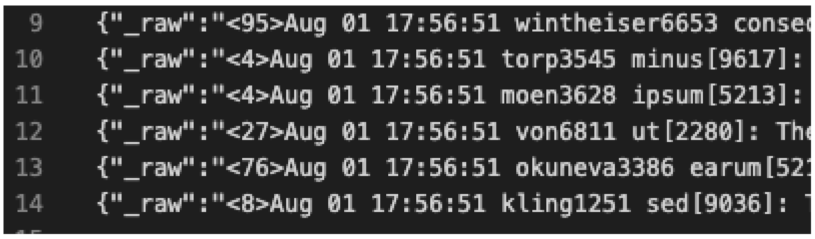
For simplicity, we'll treat the storage destination here as Amazon S3, although it could just as easily be MinIO, or any of several other options.

Choosing JSON vs. Raw Format

You can write data out of Cribl Stream in either of two formats – JSON or raw.

JSON

With this option, the parsed event, with all metadata and modifications it contains at the time it reaches the Destination step, will be wrapped in a JSON object. Each event is one line. This is newline-delimited JSON (abbreviated NDJSON). For example, here is syslog data using the JSON format option:

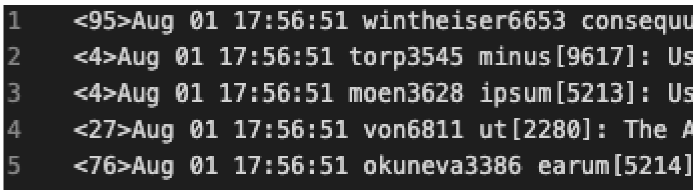


```
{"_raw":"<95>Aug 01 17:56:51 wintheiser6653 conse
{"_raw":"<4>Aug 01 17:56:51 torp3545 minus[9617]:
{"_raw":"<4>Aug 01 17:56:51 moen3628 ipsum[5213]:
{"_raw":"<27>Aug 01 17:56:51 von6811 ut[2280]: The
{"_raw":"<76>Aug 01 17:56:51 okuneva3386 earum[52
{"_raw":"<8>Aug 01 17:56:51 kling1251 sed[9036]:
```

JSON-formatted event

Raw

With this option, the contents of the event's `_raw` field – unparsed, at the time it reaches the Destination step – are written out in plaintext. Each event is one line. For example, here's the same syslog data, except unmodified and unparsed, as written out with the raw option:



```
<95>Aug 01 17:56:51 wintheiser6653 consequ
<4>Aug 01 17:56:51 torp3545 minus[9617]: Us
<4>Aug 01 17:56:51 moen3628 ipsum[5213]: Us
<27>Aug 01 17:56:51 von6811 ut[2280]: The A
<76>Aug 01 17:56:51 okuneva3386 earum[5214]
```

Raw-formatted

Which Format?

Cribl recommends using the default JSON format. Here, we expect that timestamp extractions and other vital enhancements have already been performed. Reusing that information makes sense, and will make your replay simpler.

Notice that in the screen capture above, the **raw** format simply contains the original data. There might be cases where you want this; but usually, exporting the preprocessed event is more desirable.

> Starting in version 4.3, Cribl Stream supports replaying data that has been exported as Parquet, using either the S3 Collector or the Filesystem Collector.
>
> Meanwhile, the Azure Blob Storage and Google Cloud Storage Collectors support ingesting data in Parquet format, but do not support replay.

Writing the Data Out

The Worker Nodes stage files until certain limits are reached: Time open, idle time, size, or number of files. These settings are available on the Destination configuration modal's **Advanced Settings** tab (see S3 details here).

Once any of the configured limits is reached, the Worker gzips the file and drops it into the object store. If you reach the open-file limit, the oldest file will be targeted.

The S3 Destination's settings also allow you to define how the uploaded files are partitioned. Host, time, sourcetype, source – all the metadata is available to you for this purpose. When you're replaying data from the store, these partitions will be handy, to make your replay searches faster.

We can map segments of the path back to variables (including `time`) that you can use to zero in on the exact logs you need to replay, without requiring checking `_raw`.

> For examples of the expressions Cribl recommends, see the example Destination definition below.

Retrieving the Events

With events stored in an object store, we can now point Cribl Stream to that store to Replay selected events back through the system. We want to use data in the file path as an initial level of filtering, to exclude as much data as we can from download. Object retrieval and unpacking imposes a big resource hit in the Replay process, so minimize your impact radius. Searching against `_raw` data is also possible, but should be secondary to `_time`, `sourcetype`, `index`, `host`, etc.

Retrieval details: The Leader picks one Node to do the discovery exploration, to find the potential objects that are in play. That list of targets is then doled out to the Worker Group to actually pull down the objects, and to examine them for content matches, before executing final delivery. All Worker Nodes share the workload of retrieving and re-injecting the data.

Finally, we need to process the data coming back to extract each event. As with any incoming data stream on a compatible Source, Cribl Stream can use default, or custom, Event Breaker definitions. In this case, we recommended above to use JSON as the format of the events when we write to disk, so we'll use the **Cribl** Event Breaker ruleset on this Source. This Event Breaker contains a newline-delimited-JSON definition.

Setting Up and Running Replay

With the above concepts established, let's put them to work. We'll round-trip data through our Destination and replay it in Cribl Stream.

The Store

Object storage, or any shared storage, will work. As long as all the Cribl Stream Nodes can see it, we're ready to go. For the purposes of this post, let's stick with an S3-compatible store.

You'll need all the credentials, keys, secret keys, endpoints, etc., required to access the bucket that you intend to use in your object store. (For details on cross-account access, see this blog post.) Obviously, the bucket should be able to grow to the size intended for your long-term archival needs.

Storage Class Compatibility

Cribl Stream's Collectors and Replay feature do not support the S3 Glacier, S3 Deep Glacier, or Azure archive tier, due to long retrieval times for those storage classes.

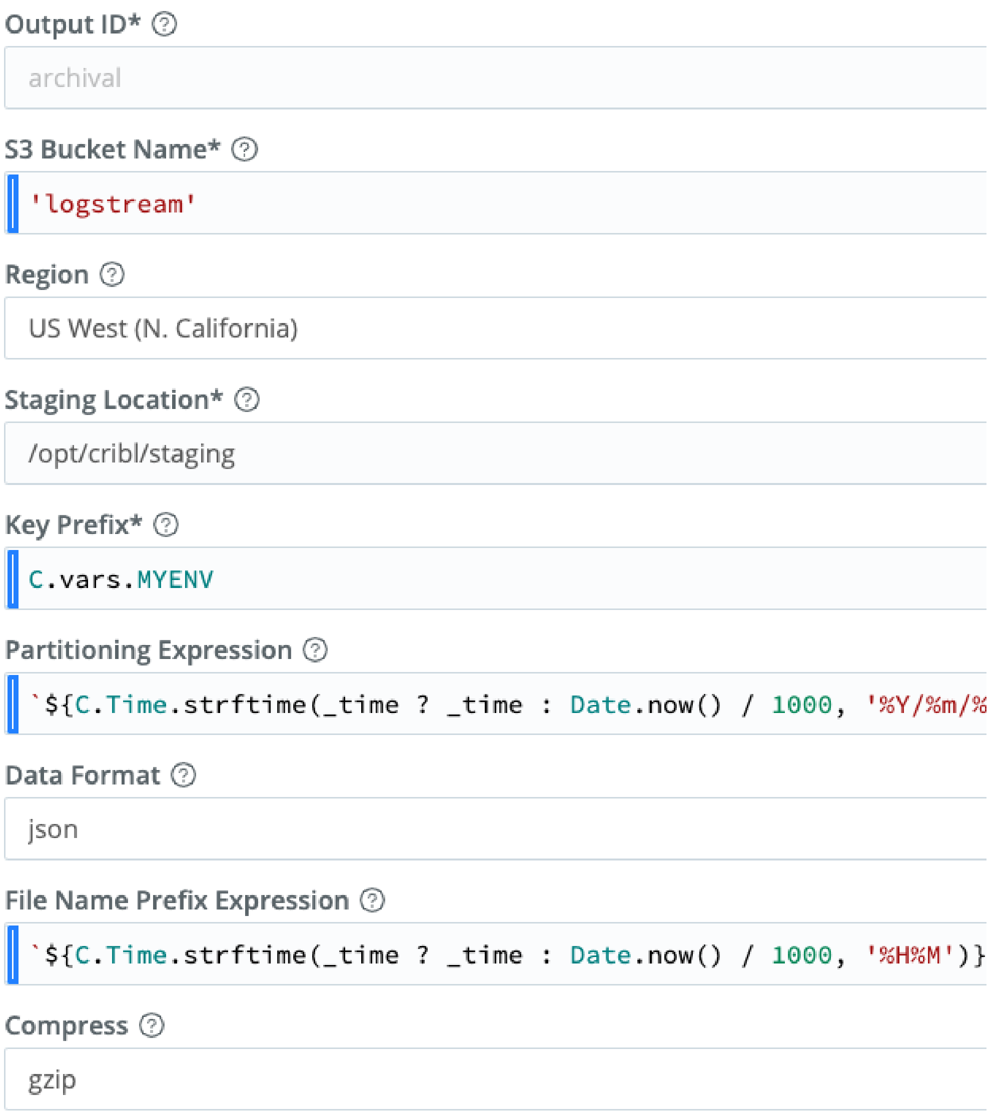
Collectors and Replay **do** support S3 Glacier Instant Retrieval when you're using the S3 Intelligent-Tiering storage class.

The Destination Definition

In your Cribl Stream Worker Group config, create a new S3 Destination. The screenshot below is an example built for this demo.

Use the Destination’s **Advanced Settings** tab to adjust the limits, if needed. The defaults are fine for most situations, but depending on your partitioning scheme, we could be talking about 100+ files. So **make sure your staging area has enough space**.

In particular, set the **Max open files** option appropriately. It overrides the size and time limits.
We recommend that the staging area be its own volume, so that you don’t fill up a more-vital volume by mistake.



Output ID*
archival

S3 Bucket Name*
```
'logstream'
```

Region
US West (N. California)

Staging Location*
/opt/cribl/staging

Key Prefix*
```
C.vars.MYENV
```

Partitioning Expression
```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%
```

Data Format
json

File Name Prefix Expression
```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%H%M')}
```

Compress
gzip

Partitioning and filename expressions

The screen capture above includes the partitioning and filename prefix expressions. Below is the full text of each expression. In a nutshell, we’re using time and other metadata to construct a path in the object store, which will be useful to us at replay time:

```
Year/Month/Day/index/host/sourcetype/HHMM-foobar.gz
```

Partitioning (one line):

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d')}/${index ? index
```

Filename:

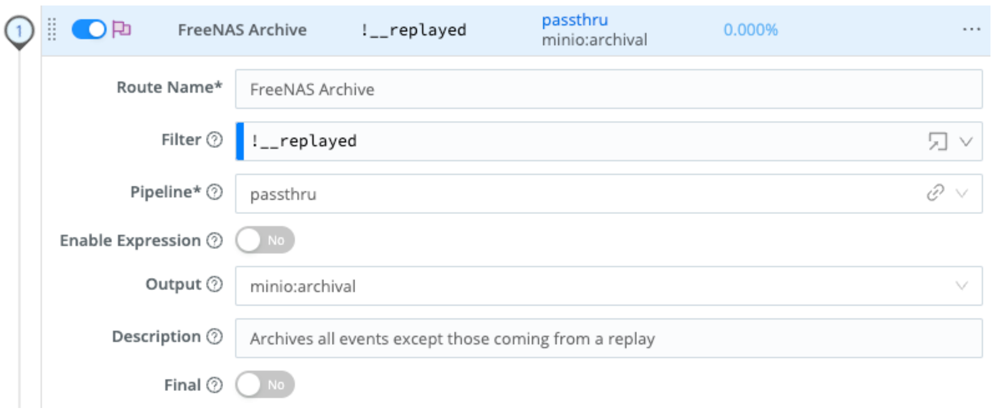
```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%H%M')}`
```

We've also included a Key Prefix from Cribl Stream's **Processing** > **Knowledge** > **Global Variables**. You could use this to partition your logs by environment, or any other qualifier.

The Archival Route

Create a new Cribl Stream Route called `Archival` that matches everything you want to archive. In this case, we've set it to `!__replayed`, and set this internal field in the Source (see the next step). Any event that is **not** coming from the defined Replay process will be archived.

Select the `passthru` Pipeline, or create an empty Pipeline and use that. Select the S3 Destination you created above. And finally, make sure **Final** is **not set**. We want data to flow through this Route, not stop at it. So, position the `Archival` Route at or near the top of the Routing table. Save your work, commit, and deploy.



Route Name*	FreeNAS Archive
Filter	!__replayed
Pipeline*	passthru
Enable Expression	No
Output	minio:archival
Description	Archives all events except those coming from a replay
Final	No

Pipeline configuration

Once saved and deployed, data should be flowing to your object store. Check your work: Go to the Monitoring dashboard and select **Data** > **Destinations** at the top. You want to see a green checkmark on the right side for your S3 Destination. If it's not green, find out why.

You can also use an S3-capable browser, like Cyberduck, to check the files more manually. (Specific troubleshooting steps are outside the scope of this doc.)

A green (healthy) destination

The Replay Collector

From the top nav of your Cribl Stream Worker Group or single instance, select **Data** > **Sources** > **Collectors** > **S3**, and create a new S3 Collector with the ID `Replay`. Use the **Auto-populate from** option to pull in the configs from your S3 Destination.

In the **Path** field, we need to establish the tokens to extract from the path on the S3 store. If you used the recommended partitioning scheme in the above example Destination definition, we recommend specifying these tokens here:

```
/${MYENV}/${_time:%Y}/${_time:%m}/${_time:%d}/${index}/${host}/${sourcetype}/${_tir
```

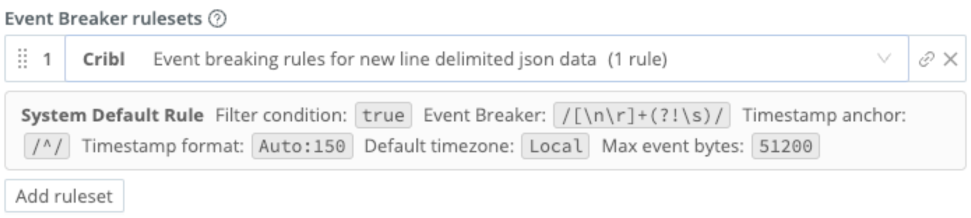
If you chose to leave out `C.Vars.MYENV`, exclude the first segment. It is **vital** that this scheme match your partitioning scheme in the Destination definition.

With these tokens defined, you can now define filters that exclude and include relevant files before Cribl Stream is required to download and open them. This cuts down on the work required, by orders of magnitude.

This is a core concept. We want to avoid having to open files to check for matches, by instead relying on key data in the path. Cribl Stream can immediately exclude, without downloading, the files that have no chance of matching our target events.

Using `_time`, `sourcetype`, `host`, and `index`, it can accurately zero in on the target files. After this high-level filtering, Cribl Stream will download and interrogate the contents of what's left, and will send the matches along to the Routing table.

Finally, under **Result Settings** > **Event Breakers**, select the `Cribl` Ruleset. This Ruleset understands how to parse the JSON packaged events stored by the Archive Destination.


Event Breaker rulesets

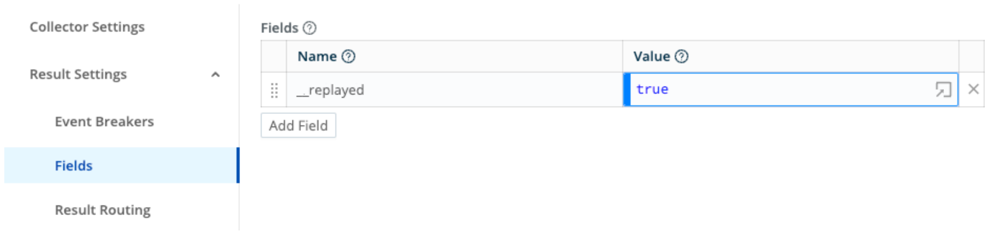
1 Cribl Event breaking rules for new line delimited json data (1 rule)

System Default Rule Filter condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp anchor: `/^/` Timestamp format: `Auto:150` Default timezone: `Local` Max event bytes: `51200`

Add ruleset

Cribl Event Breaker Ruleset

To make it easier to identify events that have been replayed, create a field (**Result Settings** > **Fields**) named `__replayed` and set it to `true`. You could filter on this field in Routes and Pipelines later. Because it's a double-underscore field, it's internal-only, and won't be passed on to your final destinations. In a previous step, we used it to prevent replayed events from being re-archived:



Fields

Name	Value
__replayed	true

__replayed field

Now save your Collector, and commit and deploy.

Testing Replay

After you've accumulated some data in your S3 store, head over to Pipelines and start a capture. For the filter, use `__replayed`, and run it for 300 seconds, 10 max events.

Once it's running, in a new browser window, navigate back to Collectors, and run your Replay Collector. Filter on `true`, and set the **Earliest** time to `-1h` and the **Latest** to `now`.

You can run it in Preview mode to make sure you get results, and then come back and do a full run. (Or you can just do a full run right off the bat if you're confident.)

With a running job, you can click on the job ID to follow its progress. You can also pop back over to the Capture browser window or tab, and you should see events there.

In a full run, the events will proceed through your Routes as normal, and will land wherever your original Routes and Pipelines dictate. In this case, they landed in Splunk, and we could easily see duplicate events – that is, **exact** duplicate events – in the 1-hour timeframe that the Collector job defined.

Once defined, this Collector can be controlled via scheduling, manual runs, or API calls. And in production use, when configuring the job's **Run configuration** or **Schedule configuration** modal, you'd want to fill in the **Filter** expression to meet your needs.

17.5.2. Using REST/API Collectors

The REST/API Endpoint Collector is powerful, but complex. This use case demonstrates several examples of building and running REST Collectors to pull data from public and simulated REST endpoints.

Check out the example REST Collector configurations in Cribl's Collector Templates repository. For many popular Collectors, the repo provides configurations (with companion Event Breakers, and event samples in some cases) that you can import into your Cribl Stream instance, saving the time you'd have spent building them yourself.

1. Basic HTTP GET

This example performs an HTTP GET operation against an external Joke API. This API uses a license key header to authenticate the user.

Discover type: None

Collect URL: `'https://matchilling-chuck-norris-jokes-v1.p.rapidapi.com/jokes/random'`

Collect parameters: None

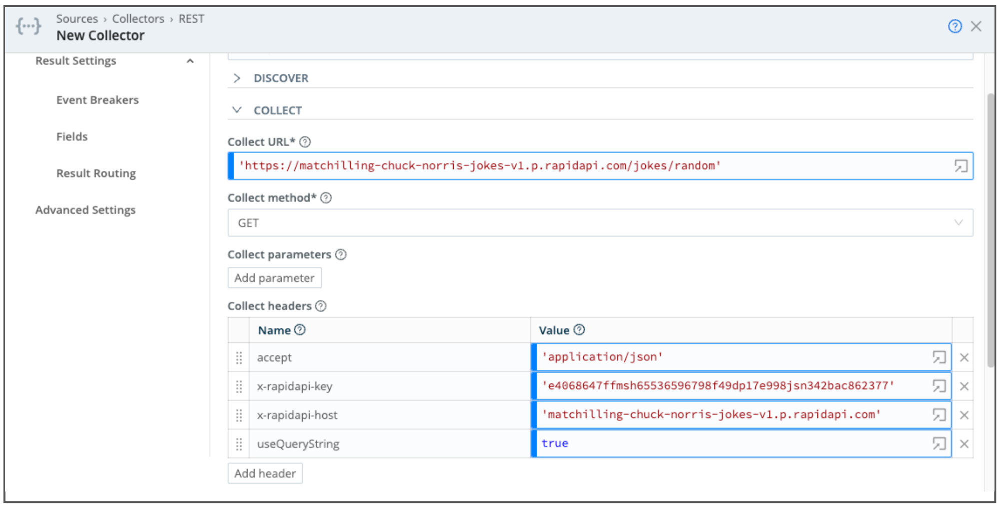
Collect headers:

```
accept: 'application/json'

x-rapidapi-key: 'e4068647ffmsh65536596798f49dp17e998jsn342bac862377'

x-rapidapi-host: 'matchilling-chuck-norris-jokes-v1.p.rapidapi.com'

useQueryString: true
```

Pagination: None

Authentication: None

Event Breaker: JSON Newline Delimited – use Cribl Stream's built-in **Cribl** > **ndjson** rule, and associate it with the Collector to parse the JSON document.



Sources › Collectors › REST
New Collector

Result Settings
Event Breakers
Fields
Result Routing
Advanced Settings

DISCOVER
COLLECT

Collect URL*
`'https://matchilling-chuck-norris-jokes-v1.p.rapidapi.com/jokes/random'`

Collect method*
GET

Collect parameters
Add parameter

Collect headers

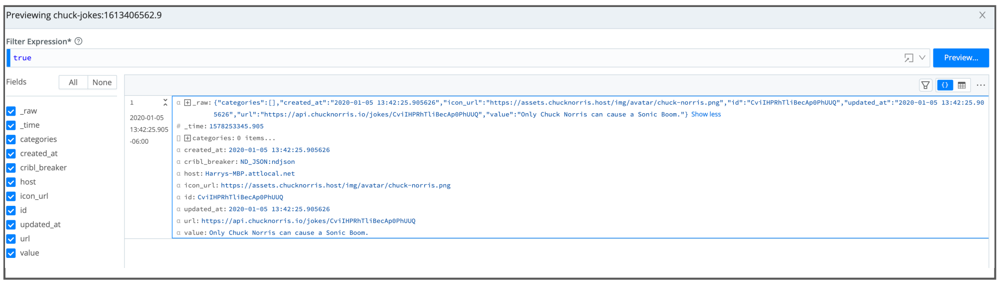
Name	Value
accept	`'application/json'`
x-rapidapi-key	`'e4068647ffmsh65536596798f49dp17e998jsn342bac862377'`
x-rapidapi-host	`'matchilling-chuck-norris-jokes-v1.p.rapidapi.com'`
useQueryString	`true`

Add header

Collector configuration for basic HTTP GET

Results

When run (in Preview mode), the Collector should return a single JSON record. If the Collector is set up with an NDJSON Event Breaker, it will look like this:



Previewing chuck-jokes:1613406562.9

Filter Expression*
`true`

Preview...

Fields: All None

- [x] _raw
- [x] _time
- [x] categories
- [x] created_at
- [x] cribl_breaker
- [x] host
- [x] icon_url
- [x] id
- [x] updated_at
- [x] url
- [x] value

```
1           _raw: {"categories":[],"created_at":"2020-01-05 13:42:25.905626","icon_url":"https://assets.chucknorris.host/img/avatar/chuck-norris.png","id":"CviIHPRhTliBecAp0PhUUQ","updated_at":"2020-01-05 13:42:25.90
                  5626","url":"https://api.chucknorris.io/jokes/CviIHPRhTliBecAp0PhUUQ","value":"Only Chuck Norris can cause a Sonic Boom."} Show less
2020-01-05  _time: 1578253345.905
13:42:25.905  categories: 0 items...
-06:00      created_at: 2020-01-05 13:42:25.905626
            cribl_breaker: ND_JSON:ndjson
            host: Harrys-MBP.attlocal.net
            icon_url: https://assets.chucknorris.host/img/avatar/chuck-norris.png
            id: CviIHPRhTliBecAp0PhUUQ
            updated_at: 2020-01-05 13:42:25.905626
            url: https://api.chucknorris.io/jokes/CviIHPRhTliBecAp0PhUUQ
            value: Only Chuck Norris can cause a Sonic Boom.
```

Returned event

2. HTTP GET with Pagination via URL Attribute

The REST Collector's Pagination feature (available in Cribl Stream 2.4.3 and above) allows collection to retrieve 1–*N* pages of data, using attributes returned in either the response body or response header. The returned attribute can either be a URL (referencing the next page), or a token that can be added to subsequent request headers or parameters.

In this example, a returned response-body attribute contains a URL that references the next page. Pagination will continue until either the Collector's **Max Pages** setting is reached, or no more pages are present (that is, the returned attribute is not present in the response body).

This example's API retrieves near-Earth asteroid data from NASA. The example uses a JSON Array Event Breaker to extract individual records from an array attribute in the response.

Discover type: None

Collect URL: `'http://www.neowsapp.com/rest/v1/neo/browse?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk'`

Collect parameters: None – Parameters in this example are added to the header. Static parameters (that is, parameters that don't reference variables) can safely be added to the URL. Any parameters that do reference variables should always be added in the **Collect parameters** section, to allow filtering of values that evaluate as undefined.

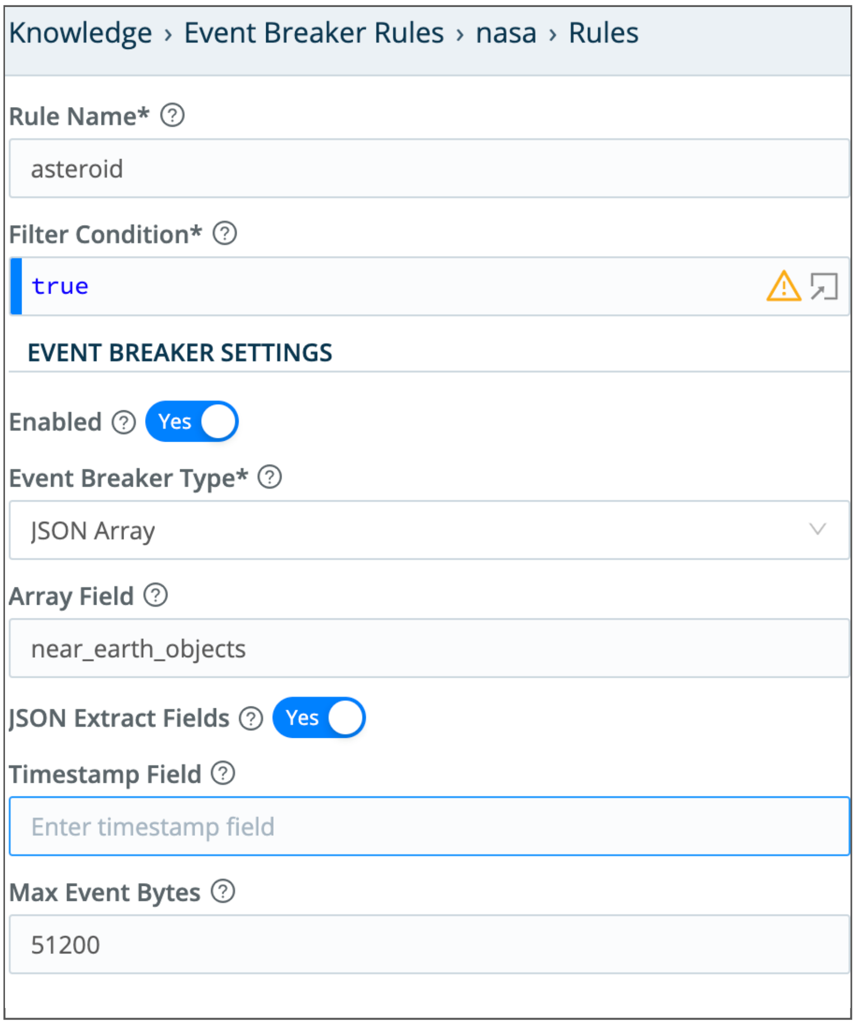
Collect headers: None

Pagination: Response Body Attribute

Response attribute: `next`

Authentication: None

Event Breaker: JSON Array


Knowledge › Event Breaker Rules › nasa › Rules

Rule Name*: asteroid

Filter Condition*: `true`

EVENT BREAKER SETTINGS

Enabled: Yes

Event Breaker Type*: JSON Array

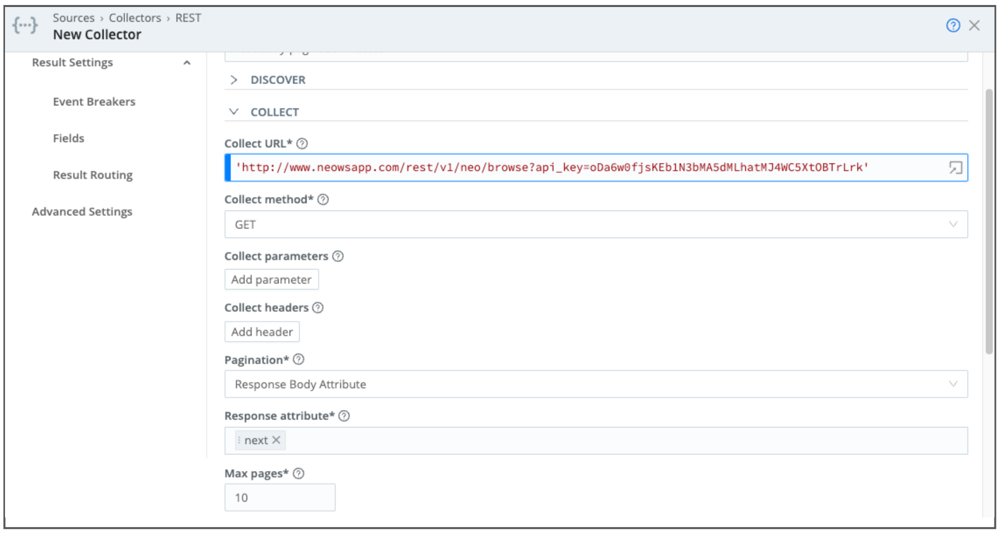
Array Field: near_earth_objects

JSON Extract Fields: Yes

Timestamp Field: Enter timestamp field

Max Event Bytes: 51200

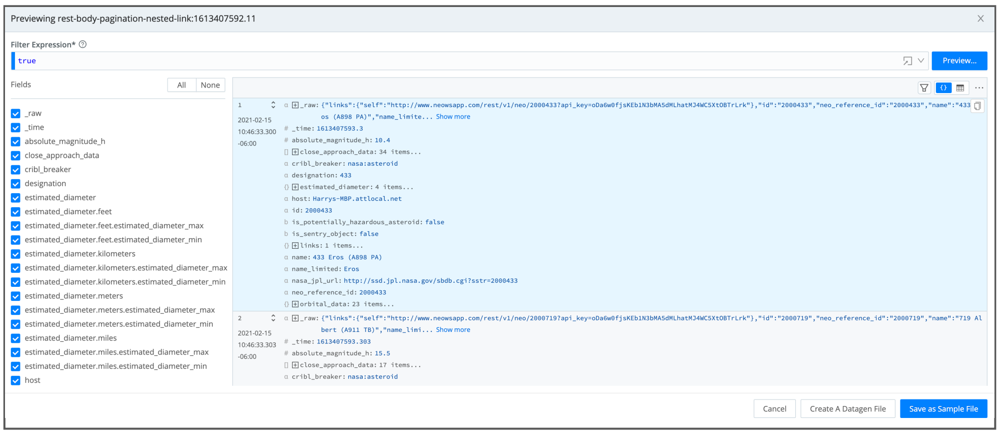
Event Breaker configuration



Collect URL*
```
'http://www.neowsapp.com/rest/v1/neo/browse?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk'
```
Collect method*: GET
Collect parameters
Collect headers
Pagination*: Response Body Attribute
Response attribute*: next
Max pages*: 10

Collector configuration for HTTP GET, paginated via URL Attribute

When run (in Preview mode), the Collector should return multiple records extracted from the Event Breaker. In this example, we limited output to 10 pages of data. This particular dataset has over 1,000 total pages, so it's a good idea to limit output to avoid a job that runs too long.



Filter Expression*: true

Fields:
- _raw
- _time
- absolute_magnitude_h
- close_approach_data
- cribl_breaker
- designation
- estimated_diameter
- estimated_diameter.feet
- estimated_diameter.feet.estimated_diameter_max
- estimated_diameter.feet.estimated_diameter_min
- estimated_diameter.kilometers
- estimated_diameter.kilometers.estimated_diameter_max
- estimated_diameter.kilometers.estimated_diameter_min
- estimated_diameter.meters
- estimated_diameter.meters.estimated_diameter_max
- estimated_diameter.meters.estimated_diameter_min
- estimated_diameter.miles
- estimated_diameter.miles.estimated_diameter_max
- estimated_diameter.miles.estimated_diameter_min
- host

```
1             _raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000433?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000433","neo_reference_id":"2000433","name":"433
                    os (A898 PA)","name_limite... Show more
2021-02-15    _time: 1613407593.3
10:46:33.300  absolute_magnitude_h: 10.4
-06:00        close_approach_data: 34 items...
              cribl_breaker: nasa:asteroid
              designation: 433
              estimated_diameter: 4 items...
              host: Harrys-MBP.attlocal.net
              id: 2000433
              is_potentially_hazardous_asteroid: false
              is_sentry_object: false
              links: 1 items...
              name: 433 Eros (A898 PA)
              name_limited: Eros
              nasa_jpl_url: http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=2000433
              neo_reference_id: 2000433
              orbital_data: 23 items...
2             _raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000719?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000719","neo_reference_id":"2000719","name":"719 Al
                    bert (A911 TB)","name_limi... Show more
2021-02-15    _time: 1613407593.303
10:46:33.303  absolute_magnitude_h: 15.5
-06:00        close_approach_data: 17 items...
              cribl_breaker: nasa:asteroid
```

Paginated events

This API allows a certain number of calls/month. Cribl recommends that you not schedule this Collector – run it ad-hoc, for testing only.

3. HTTP GET with Pagination via Response Body Attribute

This example uses Response Body Attribute pagination, which returns a token that is passed as a request parameter to retrieve subsequent pages of data. The only difference between this example and Example 2 is how the Response Body Attribute is used.

To authenticate against the GreyNoise endpoint used in this example, set up a trial account according to GreyNoise's Setting Up a Trial Account documentation.

Discover type: None

Collect URL: `'https://api.greynoise.io/v2/experimental/gnql'`

Collect method: `GET`

Collect parameters:

```
query: 'last_seen:1d'
```

```
scroll: `${scroll}`
```

Collect headers:

```
accept: 'application/json'
```

```
key: '<your-GreyNoise-API-key-here>'
```

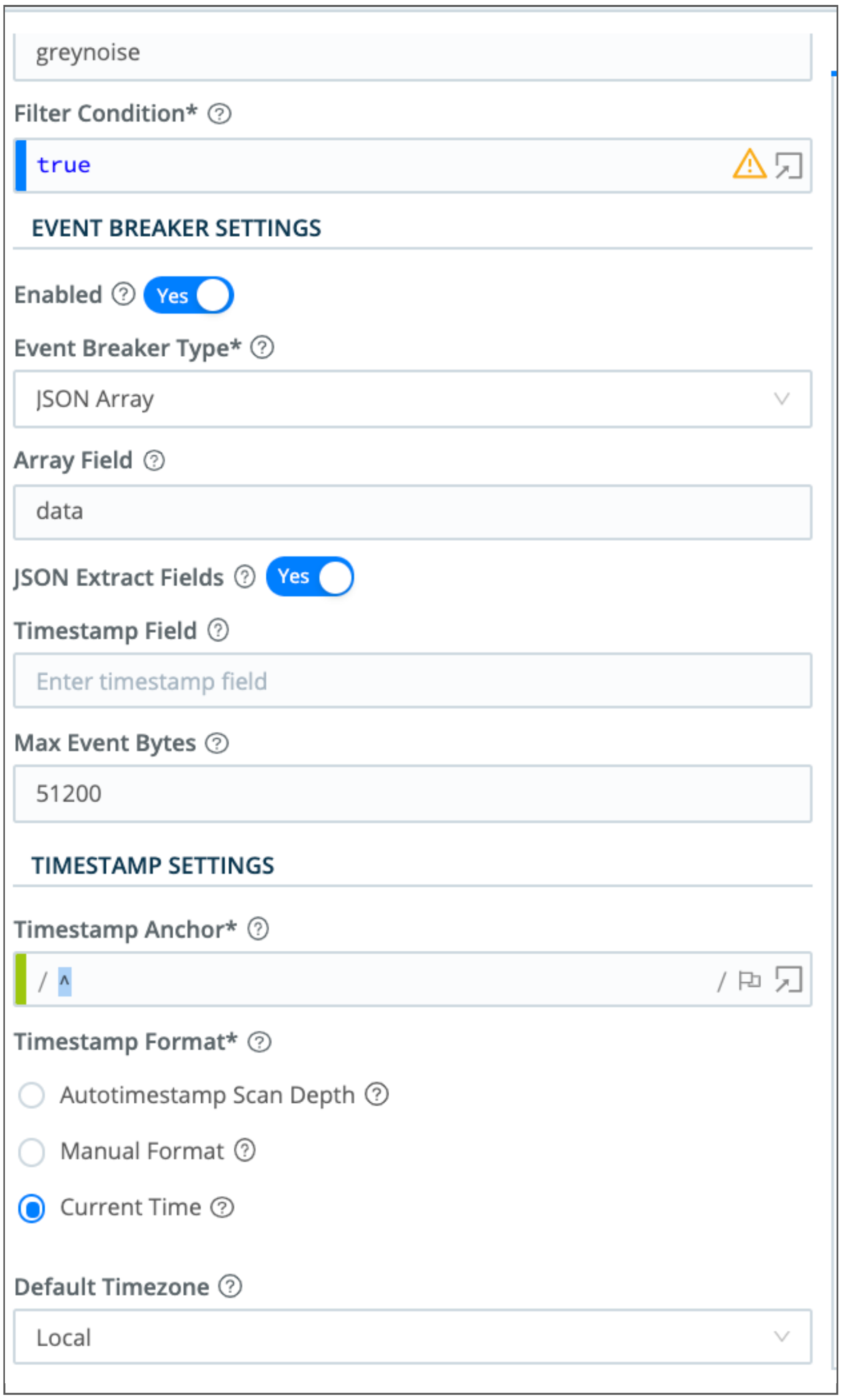
Pagination: Response Body Attribute

Response attribute: `scroll`

Max pages: `10` (or `0` to pull all data)

Authentication: None

Event Breaker: JSON Array – use the configuration shown here:



greynoise

Filter Condition*

true

EVENT BREAKER SETTINGS

Enabled: Yes

Event Breaker Type*: JSON Array

Array Field: data

JSON Extract Fields: Yes

Timestamp Field: Enter timestamp field

Max Event Bytes: 51200

TIMESTAMP SETTINGS

Timestamp Anchor*: /^/

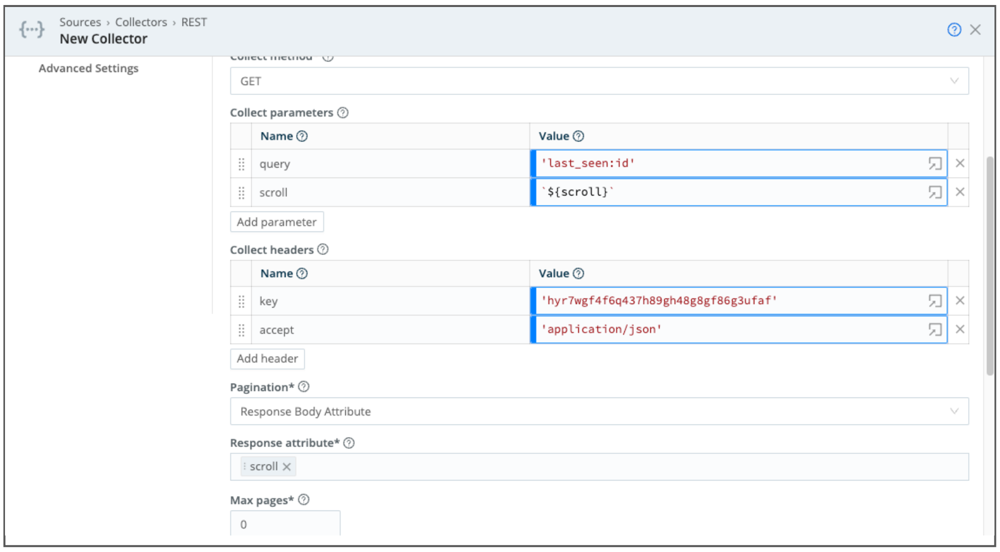
Timestamp Format*:

- Autotimestamp Scan Depth
- Manual Format
- [✓ Current Time]

Default Timezone: Local

Event Breaker configuration

In this example, the response body returns an attribute named `scroll`, which is a token that references the next page of data to fetch. We reference the attribute in **Collect parameters** using the JavaScript expression: `` `${scroll}` ``. If present, this will be passed to retrieve subsequent pages of data, until either the Collector's **Max Pages** setting is reached, or no more pages are present.


Sources › Collectors › REST
New Collector

Advanced Settings

Collect method: GET

Collect parameters

Name	Value
query	'last_seen:id'
scroll	\`${scroll}\`

Add parameter

Collect headers

Name	Value
key	'hyr7wgf4f6q437h89gh48g8gf86g3ufaf'
accept	'application/json'

Add header

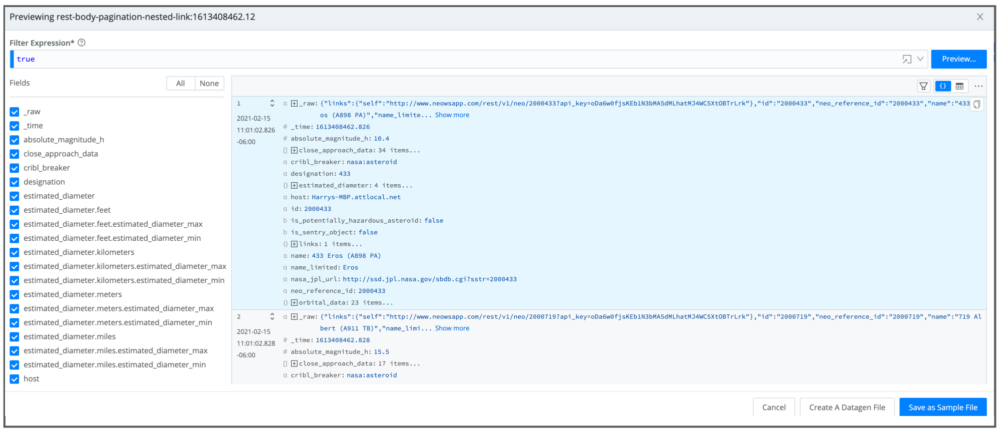
Pagination*: Response Body Attribute

Response attribute*: scroll

Max pages*: 0

Collector configuration for HTTP GET, paginated via Response Body Attribute

Collector Output


Previewing rest-body-pagination-nested-link:1613408462.12

Filter Expression*: true

Fields: _raw, _time, absolute_magnitude_h, close_approach_data, cribl_breaker, designation, estimated_diameter, estimated_diameter.feet, estimated_diameter.feet.estimated_diameter_max, estimated_diameter.feet.estimated_diameter_min, estimated_diameter.kilometers, estimated_diameter.kilometers.estimated_diameter_max, estimated_diameter.kilometers.estimated_diameter_min, estimated_diameter.meters, estimated_diameter.meters.estimated_diameter_max, estimated_diameter.meters.estimated_diameter_min, estimated_diameter.miles, estimated_diameter.miles.estimated_diameter_max, estimated_diameter.miles.estimated_diameter_min, host

```
1  2021-02-15 11:01:02.826 -06:00
_raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000433?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000433","neo_reference_id":"2000433","name":"433 Eros (A898 PA)","name_limite... Show more
_time: 1613408462.826
absolute_magnitude_h: 10.4
close_approach_data: 34 items...
cribl_breaker: nasa:asteroid
designation: 433
estimated_diameter: 4 items...
host: Harrys-MBP.attlocal.net
id: 2000433
is_potentially_hazardous_asteroid: false
is_sentry_object: false
links: 1 items...
name: 433 Eros (A898 PA)
name_limited: Eros
nasa_jpl_url: http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=2000433
neo_reference_id: 2000433
orbital_data: 23 items...

2  2021-02-15 11:01:02.828 -06:00
_raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000719?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000719","neo_reference_id":"2000719","name":"719 Albert (A911 TB)","name_limi... Show more
_time: 1613408462.828
absolute_magnitude_h: 15.5
close_approach_data: 17 items...
cribl_breaker: nasa:asteroid
```

Cancel | Create A Datagen File | Save as Sample File

Paginated events

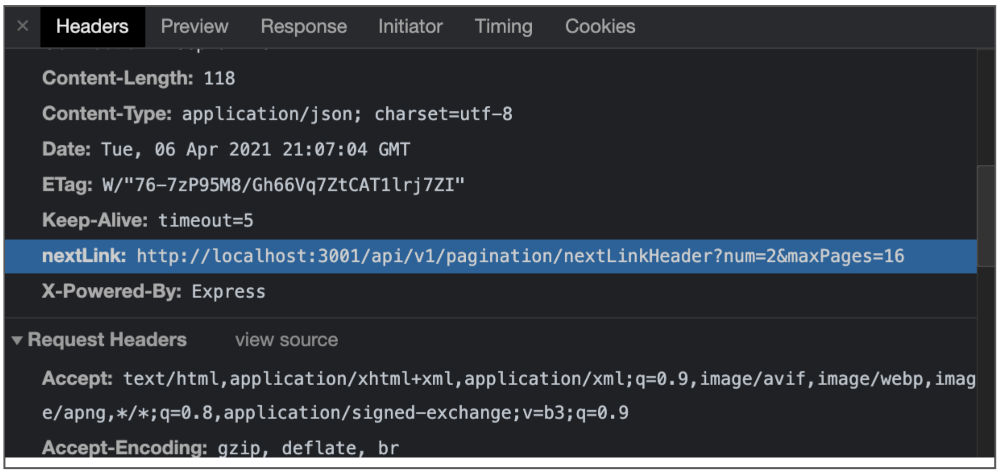
This API allows a certain number of calls/month. Cribl recommends that you not schedule this Collector – run it ad-hoc, for testing only.

For a more detailed use case around this particular API, see Cribl's Enrichment at Scale! blog post.

4. HTTP GET with Pagination via Response Header URL

This example leverages pagination using a Response Header Attribute value. The value returned can be either a URL (of the next page) or a token value (a request attribute that is passed to retrieve the next page of data).

This example is based around a local Web server on port 3001. The server returns a response header when another page of data is available, and the header contains the URL of the next page. Here's how the header looks in developer tools:



```
Content-Length: 118
Content-Type: application/json; charset=utf-8
Date: Tue, 06 Apr 2021 21:07:04 GMT
ETag: W/"76-7zP95M8/Gh66Vq7ZtCAT1lrj7ZI"
Keep-Alive: timeout=5
nextLink: http://localhost:3001/api/v1/pagination/nextLinkHeader?num=2&maxPages=16
X-Powered-By: Express
▾ Request Headers     view source
Accept: text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,imag
e/apng,*/*;q=0.8,application/signed-exchange;v=b3;q=0.9
Accept-Encoding: gzip, deflate, br
```

Next-page URL passed as Response Header Attribute

Collector Configuration

Discover type: None

Collect URL: 'http://localhost:3001/api/v1/pagination/nextLinkHeader?num=1&maxPages=16'

Collect parameters: None

Collect headers: None

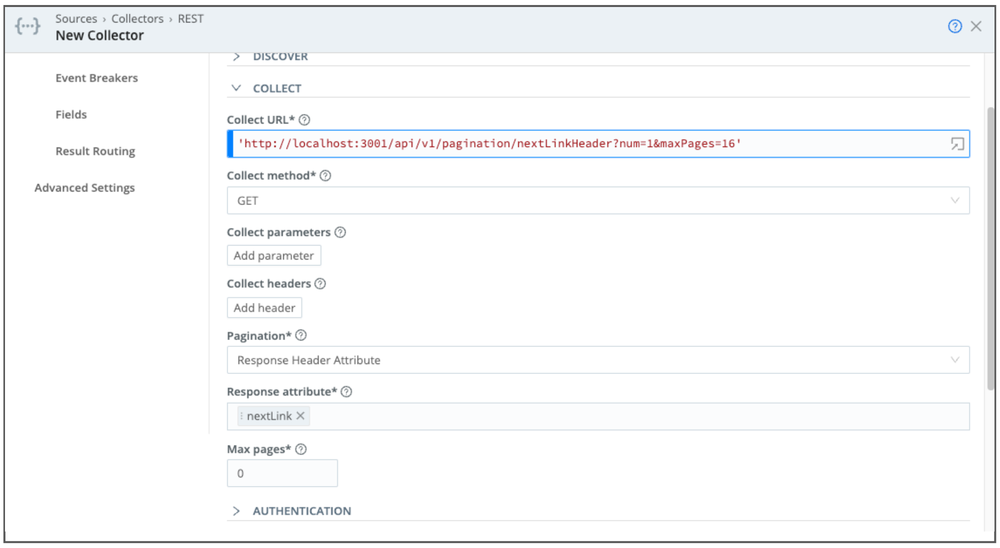
Pagination: Response Header Attribute

Response attribute: `nextLink`

Authentication: None

Event Breaker: None

You can modify the `maxPages` URL parameter to control how many pages this call returns.



Sources › Collectors › REST
New Collector

Collect URL*: `'http://localhost:3001/api/v1/pagination/nextLinkHeader?num=1&maxPages=16'`

Collect method*: GET

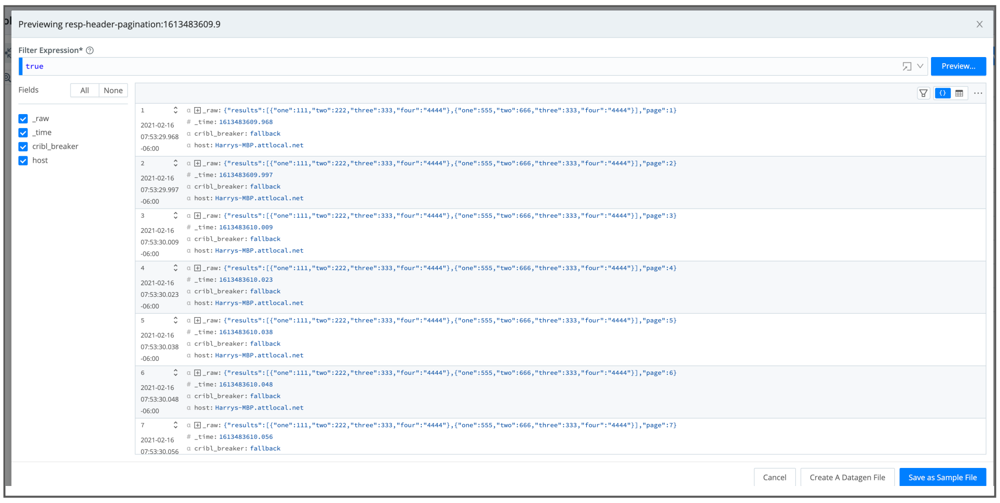
Pagination*: Response Header Attribute

Response attribute*: nextLink

Max pages*: 0

Collector configuration for HTTP GET, paginated via Response Header URL

Collector Output



Previewing resp-header-pagination:1613483609.9

Filter Expression*: `true`

```
1                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":1}
2021-02-16               _time: 1613483609.968
07:53:29.968             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
2                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":2}
2021-02-16               _time: 1613483609.997
07:53:29.997             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
3                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":3}
2021-02-16               _time: 1613483610.009
07:53:30.009             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
4                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":4}
2021-02-16               _time: 1613483610.023
07:53:30.023             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
5                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":5}
2021-02-16               _time: 1613483610.038
07:53:30.038             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
6                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":6}
2021-02-16               _time: 1613483610.048
07:53:30.048             cribl_breaker: fallback
-06:00                   host: Harrys-MBP.attlocal.net
7                        _raw: {"results":[{"one":111,"two":222,"three":333,"four":"4444"},{"one":555,"two":666,"three":333,"four":"4444"}],"page":7}
2021-02-16               _time: 1613483610.056
07:53:30.056             cribl_breaker: fallback
```

Paginated events

5. HTTP Discover and Collect with Login Authentication

In some cases, you must run an HTTP Request discovery to identify the items to collect. This example will do the following:

1. Perform a Login (POST with body containing the login credentials), to obtain an auth token that will passed in the Authorization header in all subsequent REST calls.
2. Run a REST call to discover items to be collected – in this case, log files.
3. For each log file discovered, collect the contents of that file.

4. We'll also demonstrate URL-encoding of a path element. You'd need to manually encode part of the URL in cases where unsafe ASCII characters might be present in the path element (for example, space, `$`, `/`, or `=`).

Discover type: `HTTP Request`

Discover URL: 'http://localhost:9000/api/v1/system/logs'

Discover method: GET

Discover parameters: None

Discover headers: None

Discover data field: `items`

Collect URL: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`

Collect method: GET

Collect parameters: None

Collect headers: None

Pagination: None

Authentication: Login

Login URL: `http://localhost:9000/api/v1/auth/login`

Username: `admin` (or other user)

Password: `admin` (or other user's corresponding password)

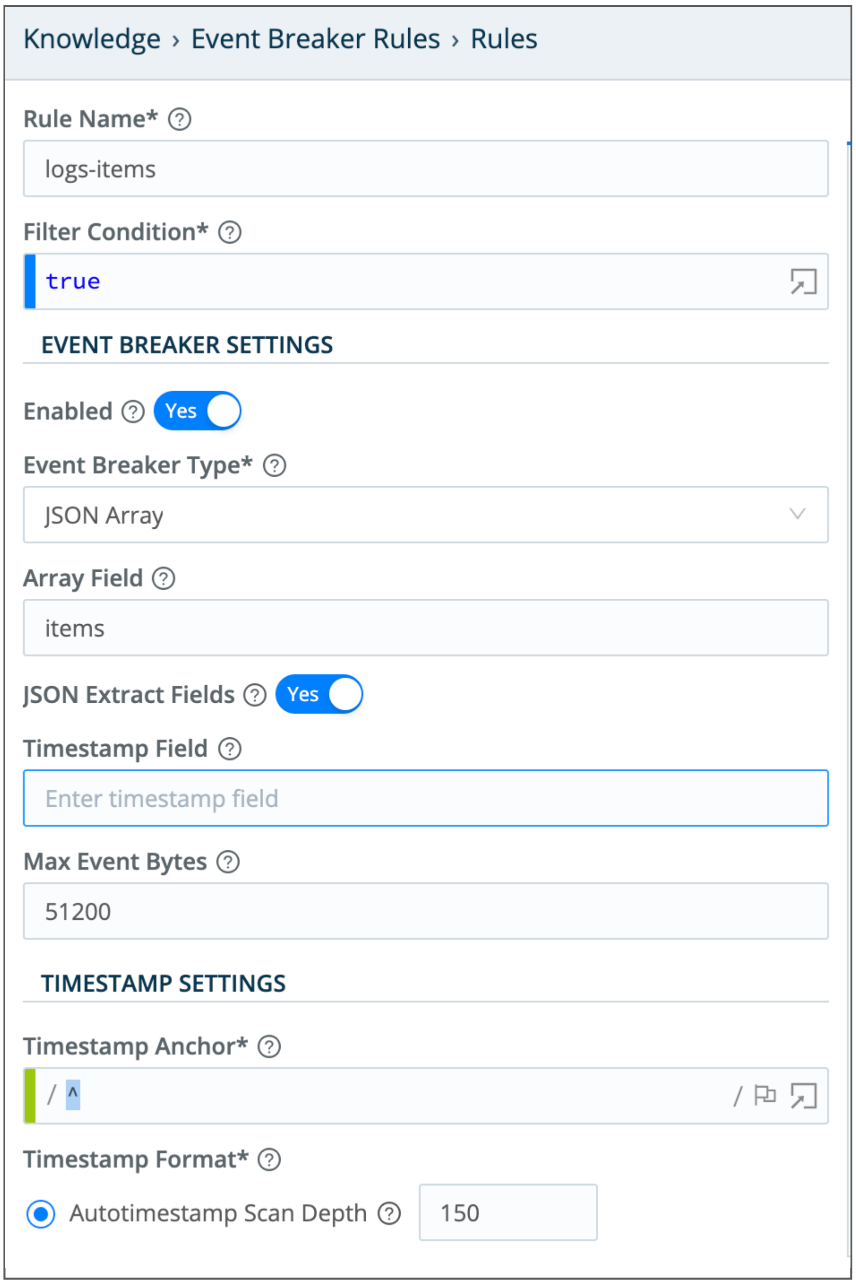
[Authentication] POST Body: ``{ "username": "${username}", "password": "${password}" }``

Token Attribute: `token`

Authorize Expression: ``Bearer ${token}``

Event Breaker: JSON Array

- **Array Field**: `items.events`


Knowledge › Event Breaker Rules › Rules

Rule Name*
logs-items

Filter Condition*
```
true
```

EVENT BREAKER SETTINGS

Enabled: Yes

Event Breaker Type*
JSON Array

Array Field
items

JSON Extract Fields: Yes

Timestamp Field
Enter timestamp field

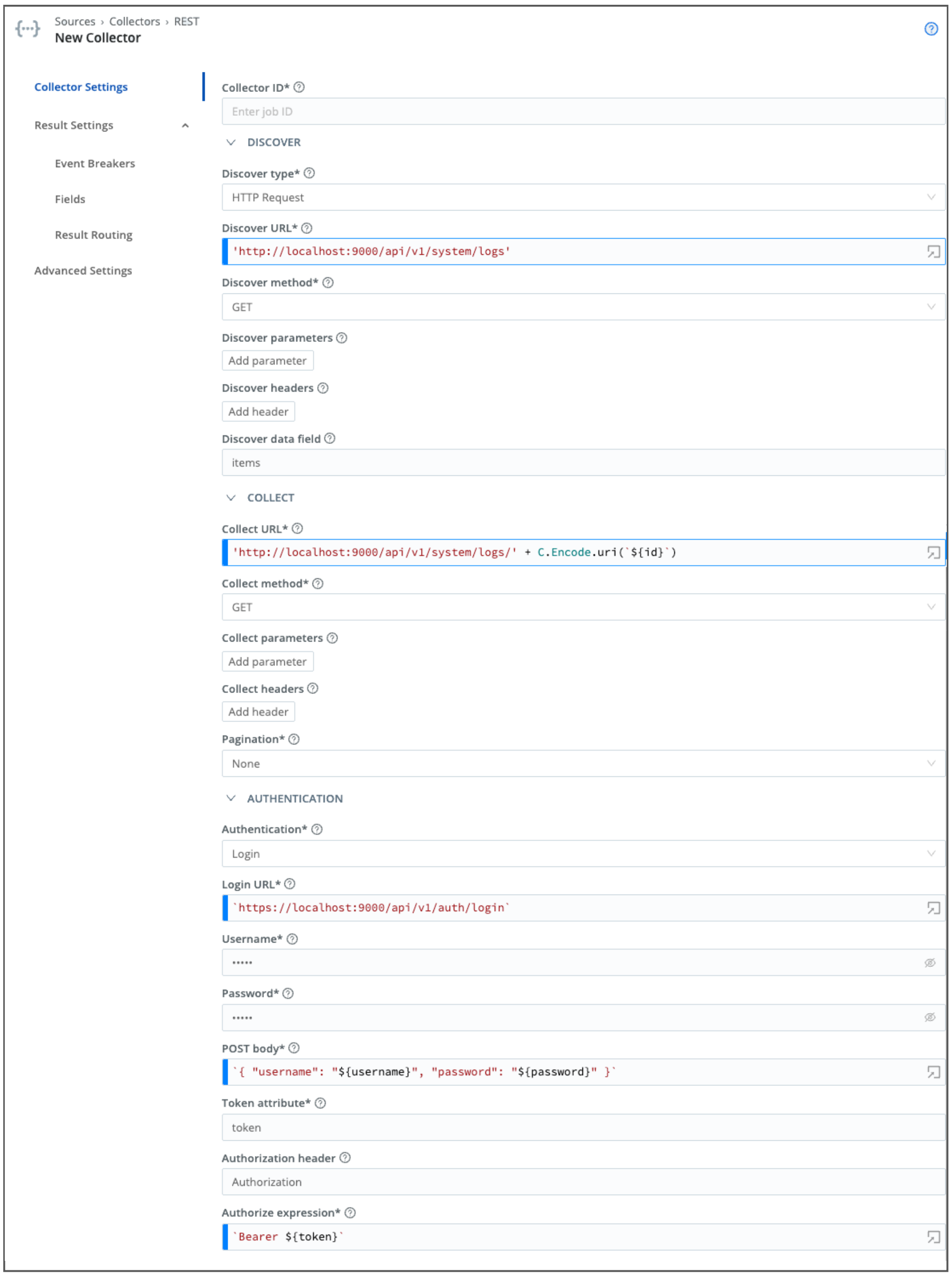
Max Event Bytes
51200

TIMESTAMP SETTINGS

Timestamp Anchor*
/ ^ /

Timestamp Format*
● Autotimestamp Scan Depth: 150

Event Breaker configuration


Sources › Collectors › REST
New Collector

Collector Settings
Result Settings
Event Breakers
Fields
Result Routing
Advanced Settings

Collector ID*: Enter job ID

DISCOVER

Discover type*: HTTP Request

Discover URL*: `'http://localhost:9000/api/v1/system/logs'`

Discover method*: GET

Discover parameters: Add parameter

Discover headers: Add header

Discover data field: items

COLLECT

Collect URL*: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`

Collect method*: GET

Collect parameters: Add parameter

Collect headers: Add header

Pagination*: None

AUTHENTICATION

Authentication*: Login

Login URL*: `` `https://localhost:9000/api/v1/auth/login` ``

Username*: •••••

Password*: •••••

POST body*: `` `{ "username": "${username}", "password": "${password}" }` ``

Token attribute*: token

Authorization header: Authorization

Authorize expression*: `` `Bearer ${token}` ``

Collector configuration for HTTP Discover and Collect with Login authentication

Login

The login call sends a POST to the login URL, passing the string derived from the POST Body JavaScript expression. Note that the variables `${username}` and `${password}` are available to this call, and are taken from the `username` and `password` text fields.

Upon successful login (`200` response code), the login token will be extracted from the response body's token attributes, as specified by the **Token Attribute** field.

Finally, the value derived from the **Authorize Expression** field will be added to the Authorization header for all subsequent calls (here, both Discover and Collect). Set this to `${token}` to reference the token obtained from the login POST request.

Discover

The Discover call here is used to discover the list of log files that can be collected. The data returned by this call has this format:

```
{
  "count": 0,
  "items": [
        {
        "id": "logFileName",
        "path": "pathToFile"
        }
  ]
}
```

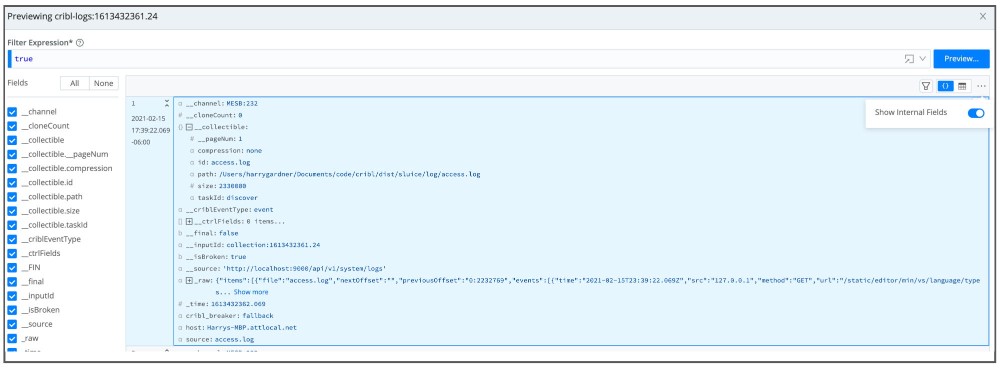
The **Discover Data Field** is used to define the array in Discover results that contains the list of items to discover. Here, each item is an object, with an attribute ID that is referenced in the Collect calls. So the Discover call generates a list of items for which Collect tasks will be created.

Collect

From the Discover task's returned list of items, each item will cause one Collect task to be created and run. An object containing the Discover item (along with some internal variables) will be passed to the Collect task.

You can reference this object's attributes as variables in the Collect task's URL, request parameters, and request headers. When running a preview, you can see the object's contents in the `__collectible` internal variable. (Enable **Show Internal Fields**, and expand `__collectible` to view the variables available).

For example, here's one of the events returned by this example's Collect operation. The `__collectible` attribute contains details identifying the page number and the URL used to obtain the data:



```
true
```

- [✓] __channel
- [✓] __cloneCount
- [✓] __collectible
- [✓] __collectible.__pageNum
- [✓] __collectible.compression
- [✓] __collectible.id
- [✓] __collectible.path
- [✓] __collectible.size
- [✓] __collectible.taskId
- [✓] __criblEventType
- [✓] __ctrlFields
- [✓] __FIN
- [✓] __final
- [✓] __inputId
- [✓] __isBroken
- [✓] __source
- [✓] _raw

```
1
2021-02-15
17:39:22.069
-06:00

α __channel: MESB:232
# __cloneCount: 0
{} ⊟ __collectible:
    # __pageNum: 1
    α compression: none
    α id: access.log
    α path: /Users/harrygardner/Documents/code/cribl/dist/sluice/log/access.log
    # size: 2330080
    α taskId: discover
α __criblEventType: event
[] ⊞ __ctrlFields: 0 items...
b __final: false
α __inputId: collection:1613432361.24
b __isBroken: true
α __source: 'http://localhost:9000/api/v1/system/logs'
α ⊞ _raw: {"items":[{"file":"access.log","nextOffset":"","previousOffset":"0:2232769","events":[{"time":"2021-02-15T23:39:22.069Z","src":"127.0.0.1","method":"GET","url":"/static/editor/min/vs/language/type
        s... Show more
# _time: 1613432362.069
α cribl_breaker: fallback
α host: Harrys-MBP.attlocal.net
α source: access.log
```

__collectible internal variable, expanded to show its contents

As you can see, `__collectible` contains a `__pageNum` variable, which shows which page of data the event was received in. Also, `__collectible` contains an `id` variable, available for use in the Collect operation. Here's how this variable is referenced in the Collect operation's URL:

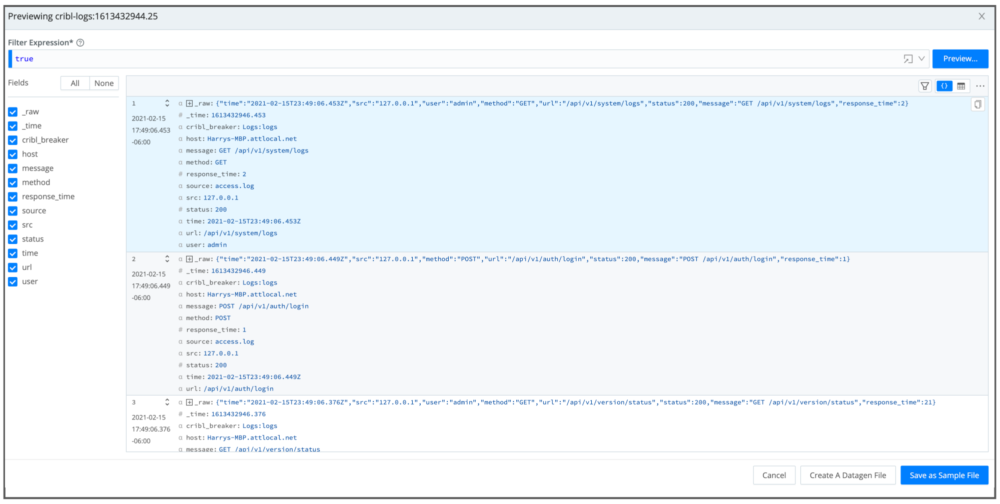
```
'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)
```

Because the variable is used in the path, and it might contain unsafe ASCII characters (specifically, space), we need to URL-encode the variable. This is the only case where the REST Collector requires URI encoding – variables that are defined directly as part of the URL. (Request parameters, not contained directly in the URL, are automatically encoded.)

The data returned by the Collect call has the following format:

```
{
  "items": [
        {
        "file": "access.log",
        "nextOffset": "",
        "previousOffset": "0:2236637",
        "events": [
        {
        "time": "2021-02-15T23:39:23.043Z",
        "src": "127.0.0.1",
        "user": "admin",
        "method": "GET",
        "url": "/api/v1/jobs/1613432361.24",
        "status": 200,
        "message": "GET /api/v1/jobs/1613432361.24",
        "response_time": 2
        },
        {
        "time": "2021-02-15T23:39:22.366Z",
        "src": "127.0.0.1",
        "user": "admin",
        "method": "GET",
        "url": "/api/v1/system/logs/worker%2F7%2Fcribl.log",
        "status": 200,
        "message": "GET /api/v1/system/logs/worker%2F7%2Fcribl.log",
        "response
...
```

The real data that we want to access is located at `items.events`. We can use a JSON Array event breaker to convert data from `events.items` into individual events that will be sent to Routes and processed by Cribl Stream. The output looks like this in Preview:



```
true
```

```
1        _raw: {"time":"2021-02-15T23:49:06.453Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/system/logs","status":200,"message":"GET /api/v1/system/logs","response_time":2}
2021-02-15  _time: 1613432946.453
17:49:06.453  cribl_breaker: Logs:logs
-06:00   host: Harrys-MBP.attlocal.net
         message: GET /api/v1/system/logs
         method: GET
         response_time: 2
         source: access.log
         src: 127.0.0.1
         status: 200
         time: 2021-02-15T23:49:06.453Z
         url: /api/v1/system/logs
         user: admin
2        _raw: {"time":"2021-02-15T23:49:06.449Z","src":"127.0.0.1","method":"POST","url":"/api/v1/auth/login","status":200,"message":"POST /api/v1/auth/login","response_time":1}
2021-02-15  _time: 1613432946.449
17:49:06.449  cribl_breaker: Logs:logs
-06:00   host: Harrys-MBP.attlocal.net
         message: POST /api/v1/auth/login
         method: POST
         response_time: 1
         source: access.log
         src: 127.0.0.1
         status: 200
         time: 2021-02-15T23:49:06.449Z
         url: /api/v1/auth/login
3        _raw: {"time":"2021-02-15T23:49:06.376Z","src":"127.0.0.1","user":"admin","method":"GET","url":"/api/v1/version/status","status":200,"message":"GET /api/v1/version/status","response_time":21}
2021-02-15  _time: 1613432946.376
17:49:06.376  cribl_breaker: Logs:logs
-06:00   host: Harrys-MBP.attlocal.net
         message: GET /api/v1/version/status
```

Collected data

If this example fails with errors of the form `statusCode: 429...Too many requests` – see Common Errors and Warnings to resolve this by relaxing the login rate limit.

6. Item List Discovery

This example demonstrates situations where the Item List discovery mechanism is useful: enabling collection based on a predefined list of items. Here, we want to collect weather information for a static list of states – each returned from Discover results as a single collection task.

Let's assume we are interested in weather for the following U.S. locations: Nashville, Dallas, and Denver. When the Discover operation runs, it will return a `__collectible` object for each location (each representing its own collection task): `{ id: ''}, {id: 'TX'}, {id: 'TN'}`.

Discover type: `Item List`

Discover items: `Nashville, Dallas, Denver`

Collect URL: `'https://community-open-weather-map.p.rapidapi.com/find'`

Collect parameters:

```
type: 'link'
```

```
units: 'imperial'
```

```
q: `${id}`
```

Collect headers:

```
x-rapidapi-host: 'community-open-weather-map.p.rapidapi.com'
```

```
x-rapidapi-key: '78934c846cmsh70cb53f75a8a54bp119d21jsn29df549b4fd6'
```

```
useQueryString: true
```

Pagination: None

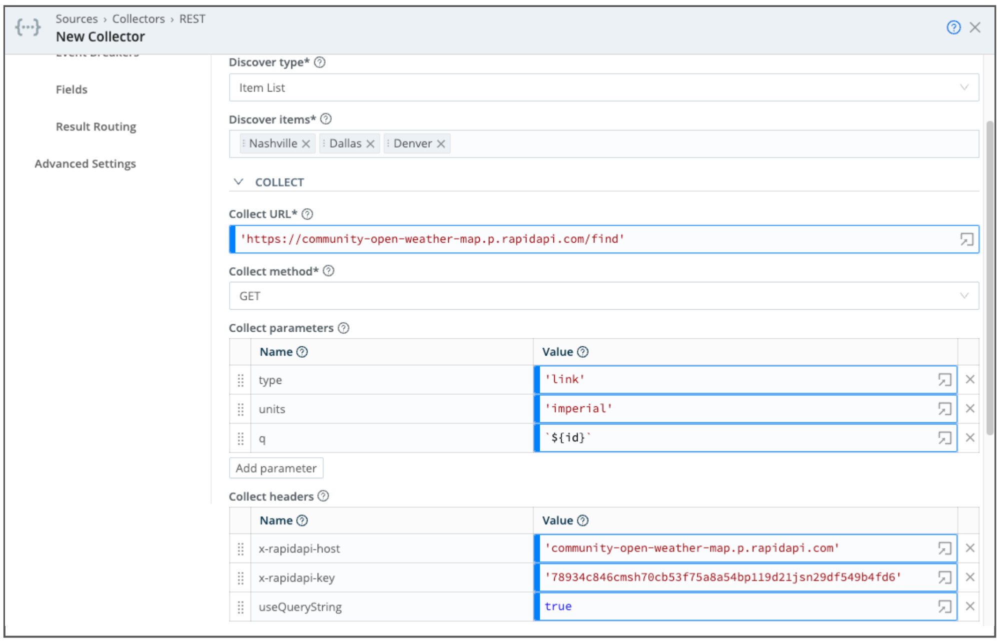
Authentication: None

Event Breaker: JSON Newline Delimited – Use a rule like **Cribl > ndjson** to parse each event and extract fields.

Fields:

```
job: weather-${__collectible.id}
```

```
city: ${__collectible.id}
```



Discover type*: Item List

Discover items*: Nashville, Dallas, Denver

Collect URL*: 'https://community-open-weather-map.p.rapidapi.com/find'

Collect method*: GET

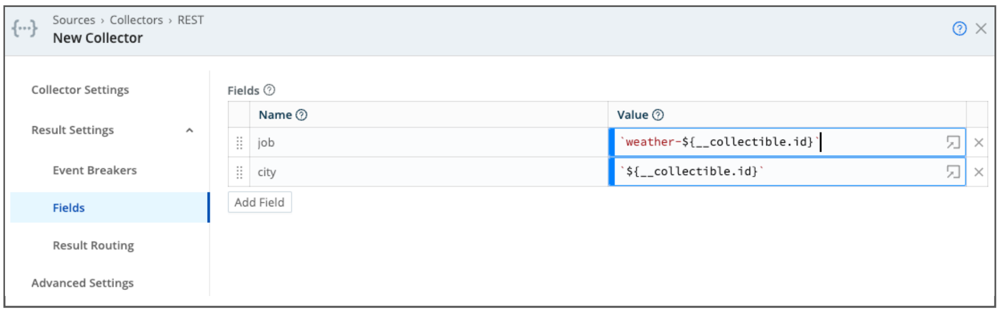
Collect parameters

Name	Value
type	'link'
units	'imperial'
q	\`${id}\`

Collect headers

Name	Value
x-rapidapi-host	'community-open-weather-map.p.rapidapi.com'
x-rapidapi-key	'78934c846cmsh70cb53f75a8a54bp119d21jsn29df549b4fd6'
useQueryString	true

Collector configuration for Discovery via Item List



Fields

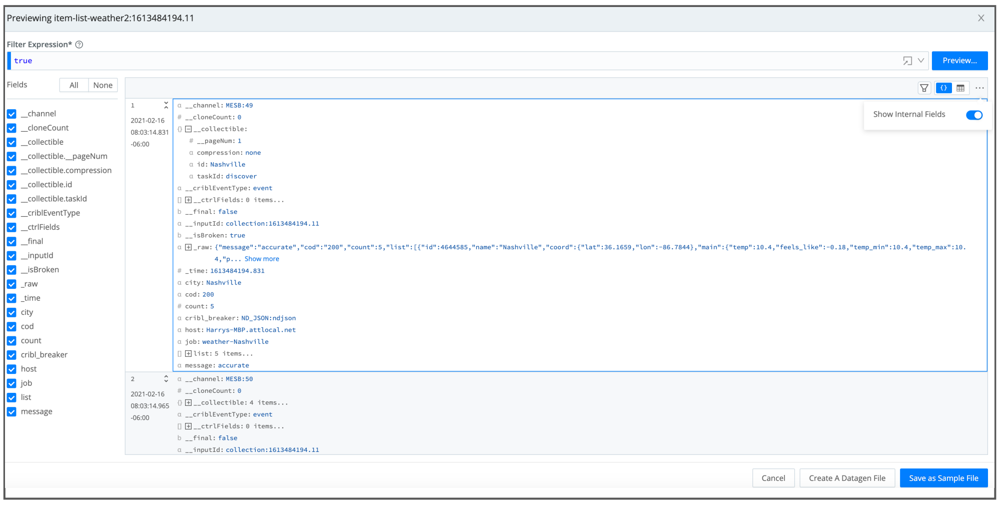
Name	Value
job	\`weather-${__collectible.id}\`
city	\`${__collectible.id}\`

Fields configuration

Collector Output

One interesting thing about this example is the addition of **Fields** to each event, using content from the internal `__collectible` attribute. This `__collectible` attribute contains results from the Discover operation, and is available in each event collected.

This demonstrates how information from the Discover operation can be transferred to events generated during the Collect operation. Note the attributes `__collectible`, `city`, and `job` in the Collector output below:



```
__channel
__cloneCount
__collectible
__collectible.__pageNum
__collectible.compression
__collectible.id
__collectible.taskId
__criblEventType
__ctrlFields
__final
__inputId
__isBroken
_raw
_time
city
cod
count
cribl_breaker
host
job
list
message
```

```
1
2021-02-16
08:03:14.831
-06:00
__channel: MESB:49
__cloneCount: 0
__collectible:
  __pageNum: 1
  compression: none
  id: Nashville
  taskId: discover
__criblEventType: event
__ctrlFields: 0 items...
__final: false
__inputId: collection:1613484194.11
__isBroken: true
_raw: {"message":"accurate","cod":"200","count":5,"list":[{"id":4644585,"name":"Nashville","coord":{"lat":36.1659,"lon":-86.7844},"main":{"temp":10.4,"feels_like":-0.18,"temp_min":10.4,"temp_max":10.
      4,"p... Show more
_time: 1613484194.831
city: Nashville
cod: 200
count: 5
cribl_breaker: ND_JSON:ndjson
host: Harrys-MBP.attlocal.net
job: weather-Nashville
list: 5 items...
message: accurate

2
2021-02-16
08:03:14.965
-06:00
__channel: MESB:50
__cloneCount: 0
__collectible: 4 items...
__criblEventType: event
__ctrlFields: 0 items...
__final: false
__inputId: collection:1613484194.11
```

Collected events

This API allows a certain number of calls/month. Cribl recommends that you not schedule this Collector – run it ad-hoc, for testing only.

7. JSON Response Discovery

Like Item List discovery, **Discover type: JSON Response** allows you to discover a predefined, static list of items. JSON Response's advantage is its ability to return an object containing more than one attribute that the Collect operation can use.

Sticking with our weather example above, imagine that we needed to use both longitude and latitude (instead of just city or state) when performing collection. This is the perfect use case for JSON Response discovery.

Discover type: `JSON Response`

Discover result: `{"items": [{"city": "Nashville", "lat": 36.174465, "lon": 86.767960}, {"city": "Dallas", "lat": 32.779167, "lon": -96.808891}, {"city": "Denver", "lat":`

`39.742043, "lon": -104.991531}] }`

Discover data field: `items`

Collect URL: `'http://api.openweathermap.org/data/2.5/weather'`

Collect headers: None

Collect parameters:

```
lat: `${lat}`
```

```
lon: `${lon}`
```

```
appid: '438d61a1db9e713240b30140e9ddfea2'
```

Pagination: None

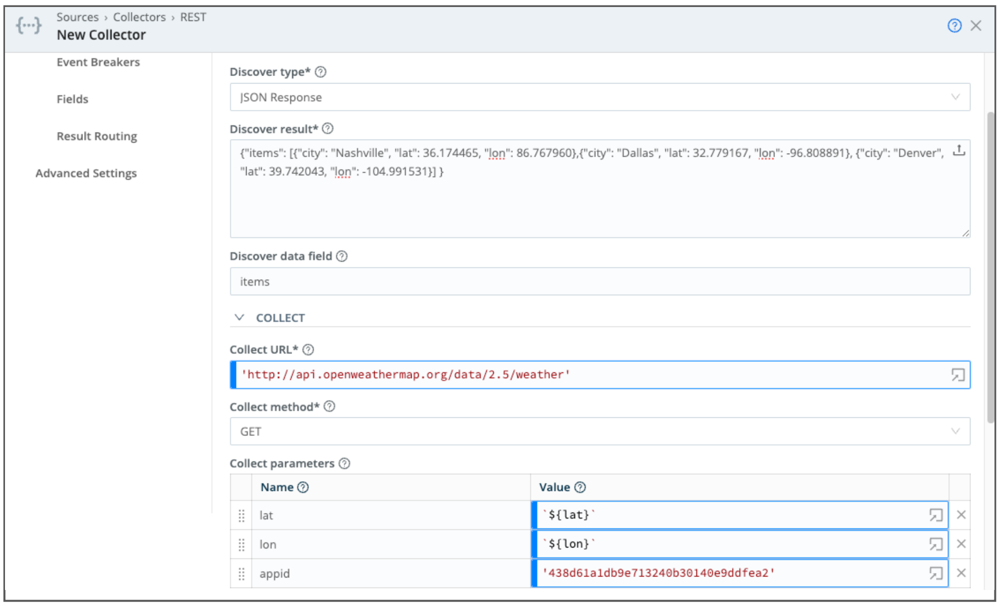
Authentication: None

Event Breaker: JSON Newline Delimited – Use a rule like **Cribl > ndjson** to parse each event and extract fields.

Fields:

```
job: `weather-${__collectible.city}`
```

```
city: `${__collectible.city}`
```



Discover type*: JSON Response

Discover result*: {"items": [{"city": "Nashville", "lat": 36.174465, "lon": 86.767960},{"city": "Dallas", "lat": 32.779167, "lon": -96.808891}, {"city": "Denver", "lat": 39.742043, "lon": -104.991531}] }

Discover data field: items

Collect URL*: 'http://api.openweathermap.org/data/2.5/weather'

Collect method*: GET

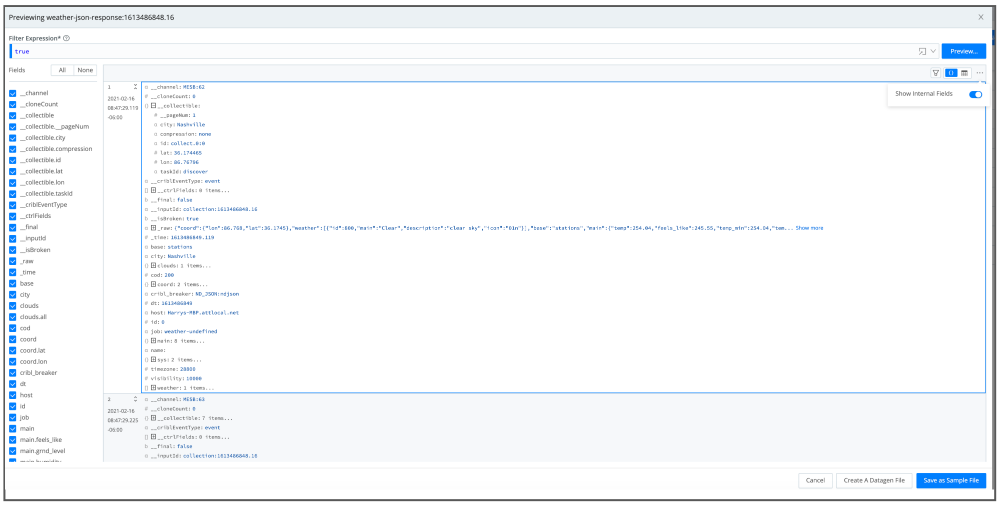
Collect parameters

Name	Value
lat	\`${lat}\`
lon	\`${lon}\`
appid	'438d61a1db9e713240b30140e9ddfea2'

Collector configuration for JSON Response Discovery

Notice how attributes present in the **Discover Result** JSON object's `items` array (`` `${lat}` ``, `` `${lon}` ``, `` `city` ``) are used in **Collect Request Parameters**, and in metadata **Fields**. Any other attribute present in the `items` array can similarly be referenced in the URL, request parameters, or request headers.

Collector Output



Filter Expression: `true`

Fields: __channel, __cloneCount, __collectible, __collectible.__pageNum, __collectible.city, __collectible.compression, __collectible.id, __collectible.lat, __collectible.lon, __collectible.taskId, __criblEventType, __ctrlFields, __final, __inputId, __isBroken, _raw, _time, base, city, clouds, clouds.all, cod, coord, coord.lat, coord.lon, cribl_breaker, dt, host, id, job, main, main.feels_like, main.grnd_level

```
1
2021-02-16
08:47:29.119
-06:00
__channel: MESB:62
__cloneCount: 0
__collectible:
  __pageNum: 1
  city: Nashville
  compression: none
  id: collect.0:0
  lat: 36.174465
  lon: 86.76796
  taskId: discover
__criblEventType: event
__ctrlFields: 0 items...
__final: false
__inputId: collection:1613486848.16
__isBroken: true
_raw: {"coord":{"lon":86.768,"lat":36.1745},"weather":[{"id":800,"main":"Clear","description":"clear sky","icon":"01n"}],"base":"stations","main":{"temp":254.04,"feels_like":245.55,"temp_min":254.04,"tem... Show more
_time: 1613486849.119
base: stations
city: Nashville
clouds: 1 items...
cod: 200
coord: 2 items...
cribl_breaker: ND_JSON:ndjson
dt: 1613486849
host: Harrys-MBP.attlocal.net
id: 0
job: weather-undefined
main: 8 items...
name:
sys: 2 items...
timezone: 28800
visibility: 10000
weather: 1 items...

2
2021-02-16
08:47:29.225
-06:00
__channel: MESB:63
__cloneCount: 0
__collectible: 7 items...
__criblEventType: event
__ctrlFields: 0 items...
__final: false
__inputId: collection:1613486848.16
```

Item List preview

This API allows a certain number of calls/month. Cribl recommends that you not schedule this Collector – run it ad-hoc, for testing only.

8. State Tracking by Latest Time

REST Collector state tracking can help prevent both duplicate data and gaps in data for subsequent collection runs. To demonstrate how to configure a Collector to use state tracking, we'll access Stream's `/system/metrics` endpoint. This guide uses an out-of-the-box Stream configuration – you may need to update endpoints and authentication parameters to match your instance's configuration.

Configure the Event Breaker

For this example we'll use an Event Breaker from Cribl's Collector Templates repository. For more information on Event Breakers, check out the documentation.

1. Click **Processing**, then **Knowledge**, then **Event Breaker Rules**.
2. Click **Add Ruleset**.
3. Click **Manage as JSON** to open a text editor view.
4. Paste in the breaker config found here.

5. Click **OK**.
6. Click **Save**.

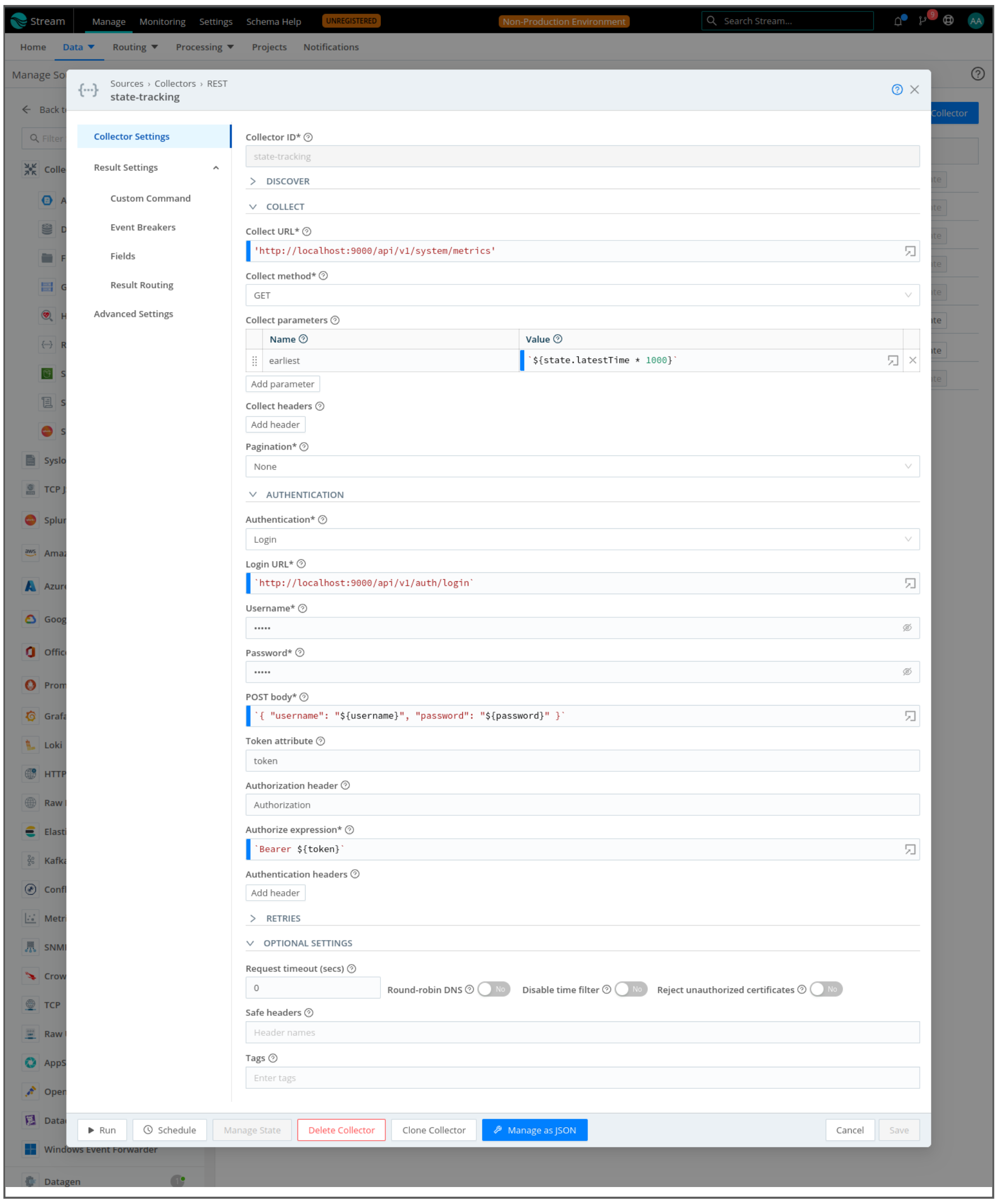
Configure the Collector

Set the following fields in the `Collector Settings` section

- **Collector ID**: `state-tracking-test`
- **Collect URL**: `'http://localhost:9000/api/v1/system/metrics'`
- **Collect parameters**:
  - **Name**: `earliest` | **Value**: `` `${state.latestTime * 1000}` ``
- **Authentication**: `Login`
- **Login URL**: `'http://localhost:9000/api/v1/auth/login'` (note that this is updated from the default `https` value)
- **Username**: `admin`
- **Password**: `admin`
- **Token attribute**: `token`

Click **Result Settings**, then **Event Breakers**. Then click **Add ruleset** and choose the `criblApi` ruleset from the earlier step.

At this point, your configuration should look like this:


Sources › Collectors › REST
state-tracking

Collector ID*: state-tracking

COLLECT

Collect URL*: `'http://localhost:9000/api/v1/system/metrics'`

Collect method*: GET

Collect parameters

Name	Value
earliest	`` `${state.latestTime * 1000}` ``

Pagination*: None

AUTHENTICATION

Authentication*: Login

Login URL*: `` `http://localhost:9000/api/v1/auth/login` ``

Username*: •••••

Password*: •••••

POST body*: `` `{ "username": "${username}", "password": "${password}" }` ``

Token attribute: token

Authorization header: Authorization

Authorize expression*: `` `Bearer ${token}` ``

OPTIONAL SETTINGS

Request timeout (secs): 0

Round-robin DNS: No

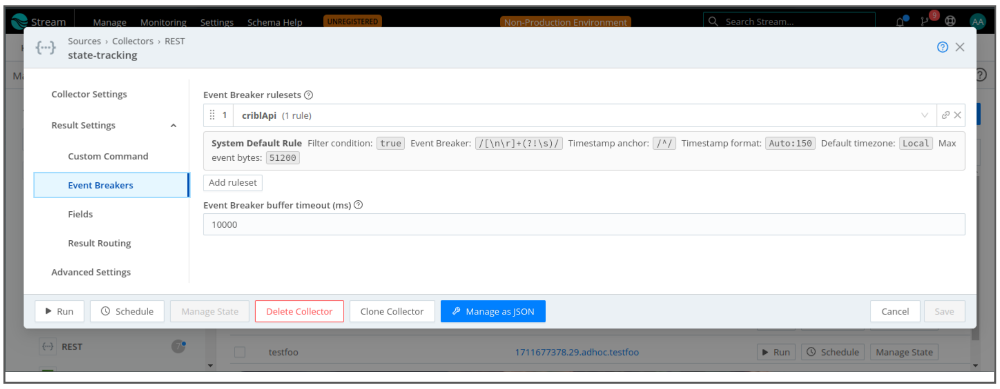
Disable time filter: No

Reject unauthorized certificates: No

Safe headers: Header names

Tags: Enter tags

Collector Config - Collector Settings


Sources › Collectors › REST
state-tracking

Event Breaker rulesets

1 criblApi (1 rule)

System Default Rule Filter condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp anchor: `/^/` Timestamp format: `Auto:150` Default timezone: `Local` Max event bytes: `51200`

Event Breaker buffer timeout (ms): 10000

Collector Config - Event Breakers

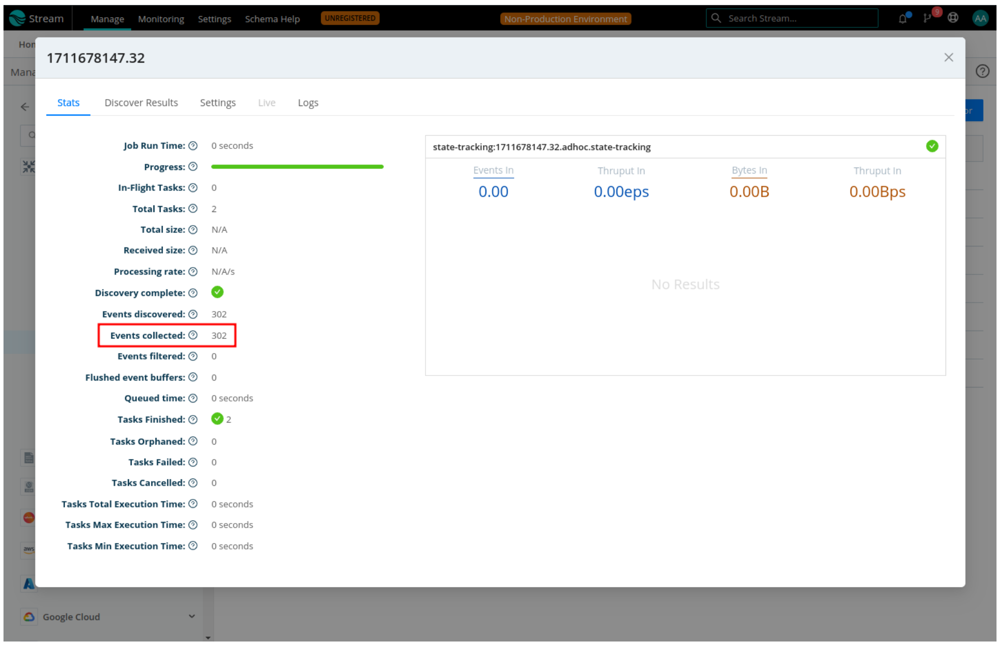
Finally, click **Save**.

Run the Collector

1. From the **Actions** column, click **Run** for your newly configured Collector.
2. Click **Full Run** to select the right collection mode. Preview and Discovery runs do not support state tracking.
3. Expand the **State Tracking** section, then set **Enabled** to `Yes`. We can use the default values for **State update expression** and **State merge expression**. For more information, see Understanding State Expression Fields.
4. Click **Run** to start the collection.

Check the Results

Since this is the first run of the Collector, there was no state value to derive `earliest` from. As a result, many events were likely returned. You can check how many by clicking the link to **Latest Ad Hoc Run** from the Collector list view. From here, check the **Events collected** result.



1711678147.32

Job Run Time:	0 seconds
Progress:	
In-Flight Tasks:	0
Total Tasks:	2
Total size:	N/A
Received size:	N/A
Processing rate:	N/A/s
Discovery complete:	✓
Events discovered:	302
Events collected:	302
Events filtered:	0
Flushed event buffers:	0
Queued time:	0 seconds
Tasks Finished:	✓ 2
Tasks Orphaned:	0
Tasks Failed:	0
Tasks Cancelled:	0
Tasks Total Execution Time:	0 seconds
Tasks Max Execution Time:	0 seconds
Tasks Min Execution Time:	0 seconds

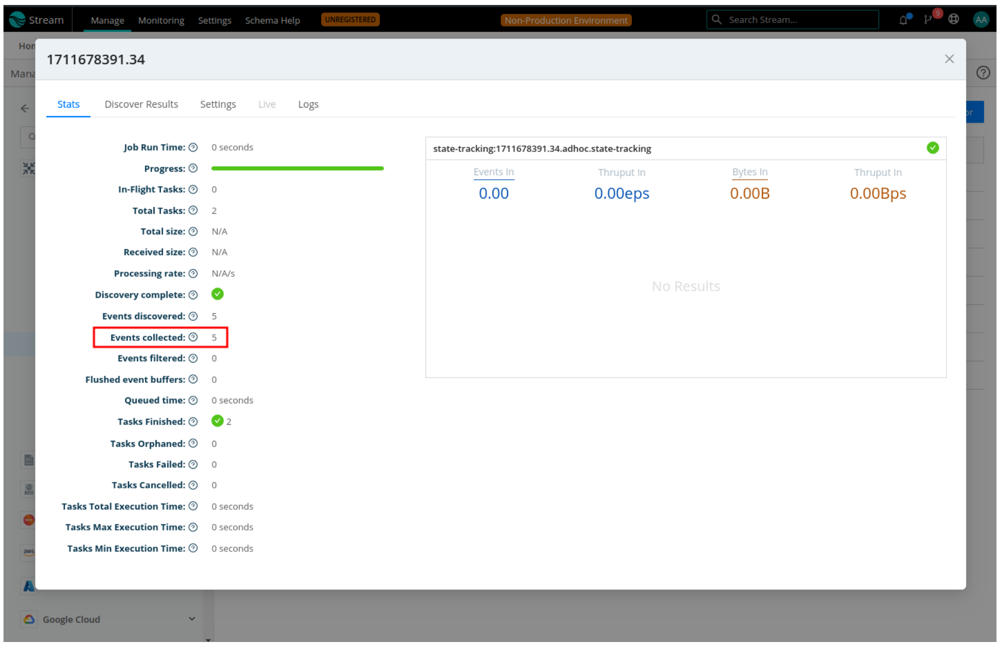
state-tracking:1711678147.32.adhoc.state-tracking

Events In	Thruput In	Bytes In	Thruput In
0.00	0.00eps	0.00B	0.00Bps

No Results

First Run Results

Now, here's where we can see the value of state tracking. Complete the steps in the **Running the Collector** section again. If you check the `Latest Ad Hoc Run` results again, you'll see that fewer events were collected. This is because the `earliest` parameter resolved to the latest time from the first run, meaning that we only picked up new metrics that were generated since the last run!



1711678391.34

Job Run Time:	0 seconds
Progress:	
In-Flight Tasks:	0
Total Tasks:	2
Total size:	N/A
Received size:	N/A
Processing rate:	N/A/s
Discovery complete:	✓
Events discovered:	5
Events collected:	5
Events filtered:	0
Flushed event buffers:	0
Queued time:	0 seconds
Tasks Finished:	✓ 2
Tasks Orphaned:	0
Tasks Failed:	0
Tasks Cancelled:	0
Tasks Total Execution Time:	0 seconds
Tasks Max Execution Time:	0 seconds
Tasks Min Execution Time:	0 seconds

state-tracking:1711678391.34.adhoc.state-tracking

Events In	Thruput In	Bytes In	Thruput In
0.00	0.00eps	0.00B	0.00Bps

No Results

Second Run Results

17.5.3. Lacework API Collection

You can configure the Lacework v2 API within Stream. This enables you to collect data from the Lacework API without introducing custom scripts or add-ons into your Lacework environment.

Your workflow will be based on the Discover and Collect pattern that's standard for REST/API Collectors. In the Lacework API variation described here, the Discover job will generate the access token that the collection job uses in requests to the API. As an example, we'll configure Cribl Stream to collect Host Vulnerability data, using optional filters available in the Lacework API.

Before you begin, make sure you have a Lacework API `KeyID` and `secretKey`, or create them as described in the Lacework docs.

Configuring the REST Collector

Collector Sources currently cannot be selected or enabled in the QuickConnect UI.

For additional details about all the configuration options specified here, see our REST/API Endpoint and Scheduling and Running topics.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Data Sources** page's tiles or the **Sources** left nav. Click **Add Collector** to open the **REST** > **New Collector** modal. Enter a **Collector ID**, then complete the following options and fields.

Discover Settings

From the **Discover type** drop-down, select `HTTP Request`. Then complete the **Discover** settings as follows.

Discover URL: Enter the URL at which Cribl Stream should access the Lacework `v2/access/tokens` endpoint.

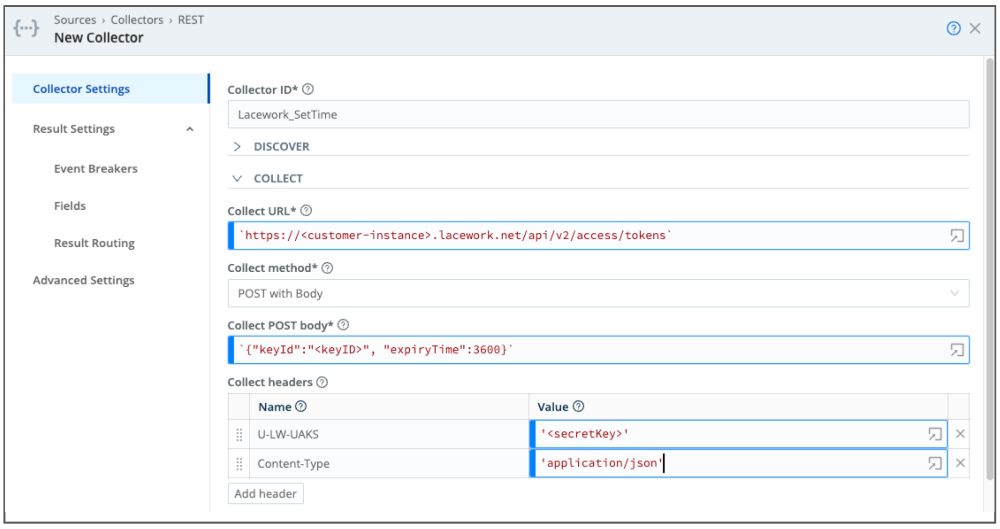
Discover method: From the drop-down, select `POST with Body`.

Discover POST Body: Enter `` `{"keyId":"<keyID>", "expiryTime":3600}` ``, substituting your Lacework API `KeyID` for the placeholder.

Discover headers: Create the two headers specified in the table below, substituting your Lacework API `secretKey` for the placeholder.

Name	Value
`U-LW-UAKS`	`'<secretKey>'`
`Content-Type`	`'application/json'`

Your Collector configuration should look similar to this:


Sources › Collectors › REST
New Collector

Collector Settings
Result Settings
Event Breakers
Fields
Result Routing
Advanced Settings

Collector ID*: Lacework_SetTime
DISCOVER
COLLECT
Collect URL*: `` `https://<customer-instance>.lacework.net/api/v2/access/tokens` ``
Collect method*: POST with Body
Collect POST body*: `` `{"keyId":"<keyID>", "expiryTime":3600}` ``
Collect headers

Name	Value
U-LW-UAKS	`'<secretKey>'`
Content-Type	`'application/json'`

Add header

Discover Settings

Collect Settings

Collect URL: Enter the URL at which Cribl Stream should access the Lacework `v2/Vulnerabilities/Hosts/search` endpoint.

Collect method: From the drop-down, select `POST with Body`.

Collect POST body: Enter the following request body.

```
`{"timeFilter": { "startTime": "${C.Time.strftime(earliest || new Date().getTime(
```

Here's what's happening in the request body example:

In our example, we've chosen to collect data only about hosts where one particular vulnerability, `CVE-2018-11233`, was detected. To accomplish this, we've added a `filters` element that specifies that the value of the `vulnId` field must equal `CVE-2018-11233`.

Note the `earliest` and `latest` variables. Later, in the **Run Collector** modal, you'll set **Time Range** values that will populate these variables when the Collector runs. Both variables are formatted as UNIX epoch time, in seconds units. (When using them in contexts that require milliseconds resolution, multiply them by 1,000 to convert to ms.) If you omit these variables, jobs will run for a period of 24 hours.

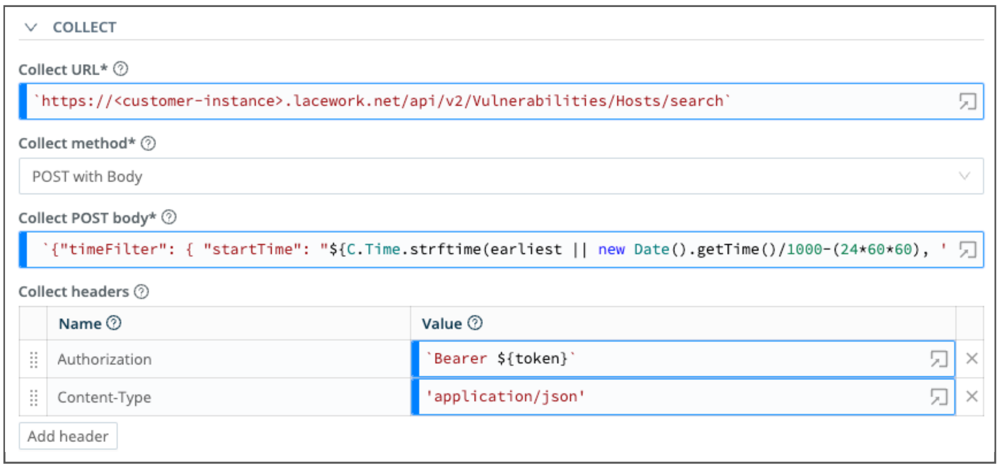
Collect Headers

We know that the Discover job, which we will run before the Collect job, will make a request to the Lacework API `v2/access/tokens` endpoint, and that the Lacework API's response body will include an access token. That means that the access token will be available to the Collect job, to pass in a Collect header.

Create the two **Collect header**s specified in the table below. The first will convey the access token.

Name	Value
`Authorization`	`` `Bearer ${token}` ``
`Content-Type`	`'application/json'`

Your **Collect** settings should look similar to this:


COLLECT

Collect URL*
`` `https://<customer-instance>.lacework.net/api/v2/Vulnerabilities/Hosts/search` ``

Collect method*
POST with Body

Collect POST body*
```
`{"timeFilter": { "startTime": "${C.Time.strftime(earliest || new Date().getTime()/1000-(24*60*60), '
```

Collect headers

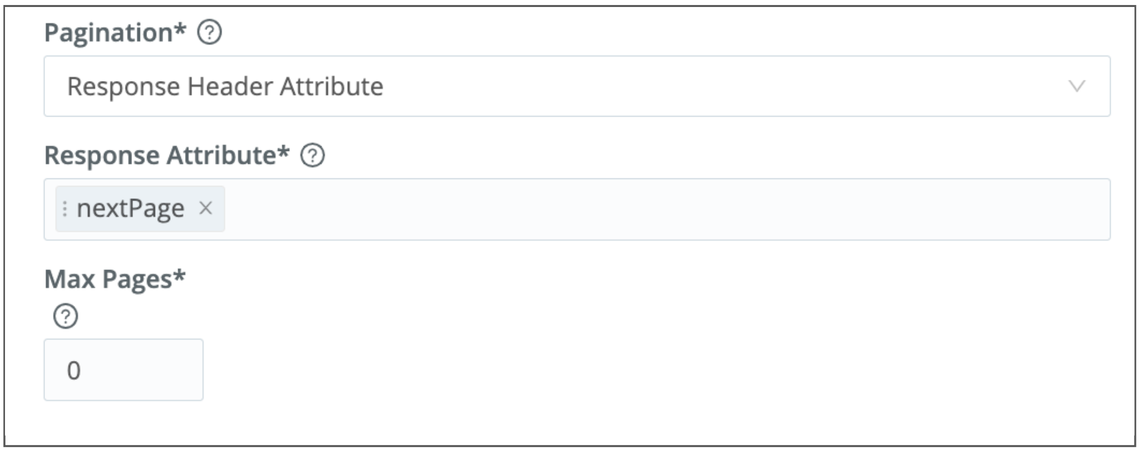
Name	Value
Authorization	`` `Bearer ${token}` ``
Content-Type	`'application/json'`

Adding a Collect header

Pagination

From the **Pagination** drop-down, select `Response Header Attribute`. In the **Response Attribute** field, enter `nextPage`. This configures your Collector to work with the Lacework API response body, which includes nested fields of pagination metadata, such that `urls` contains `nextPage`, whose value is the Next Page URL.

You should see something like this:


Pagination* ⓘ

Response Header Attribute

Response Attribute* ⓘ

nextPage

Max Pages* ⓘ

0

Configuring pagination

The Lacework API authentication mechanism requires HTTP header parameters. Since Cribl Stream does not (currently) support header parameters for authentication, we cannot use that authentication method, and you should skip the **Authentication** settings. This is why we use the Discover job to obtain an access token, and then include that token in the Collect job.

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.

Result Settings

Every call to the Lacework API `v2/Vulnerabilities/Hosts/search` endpoint returns data formatted as a single event. Every event contains multiple nested JSON arrays, where each array is a `data` element. We'll now create an Event Breaker that parses the larger structure into individual events – one for each `data` element.

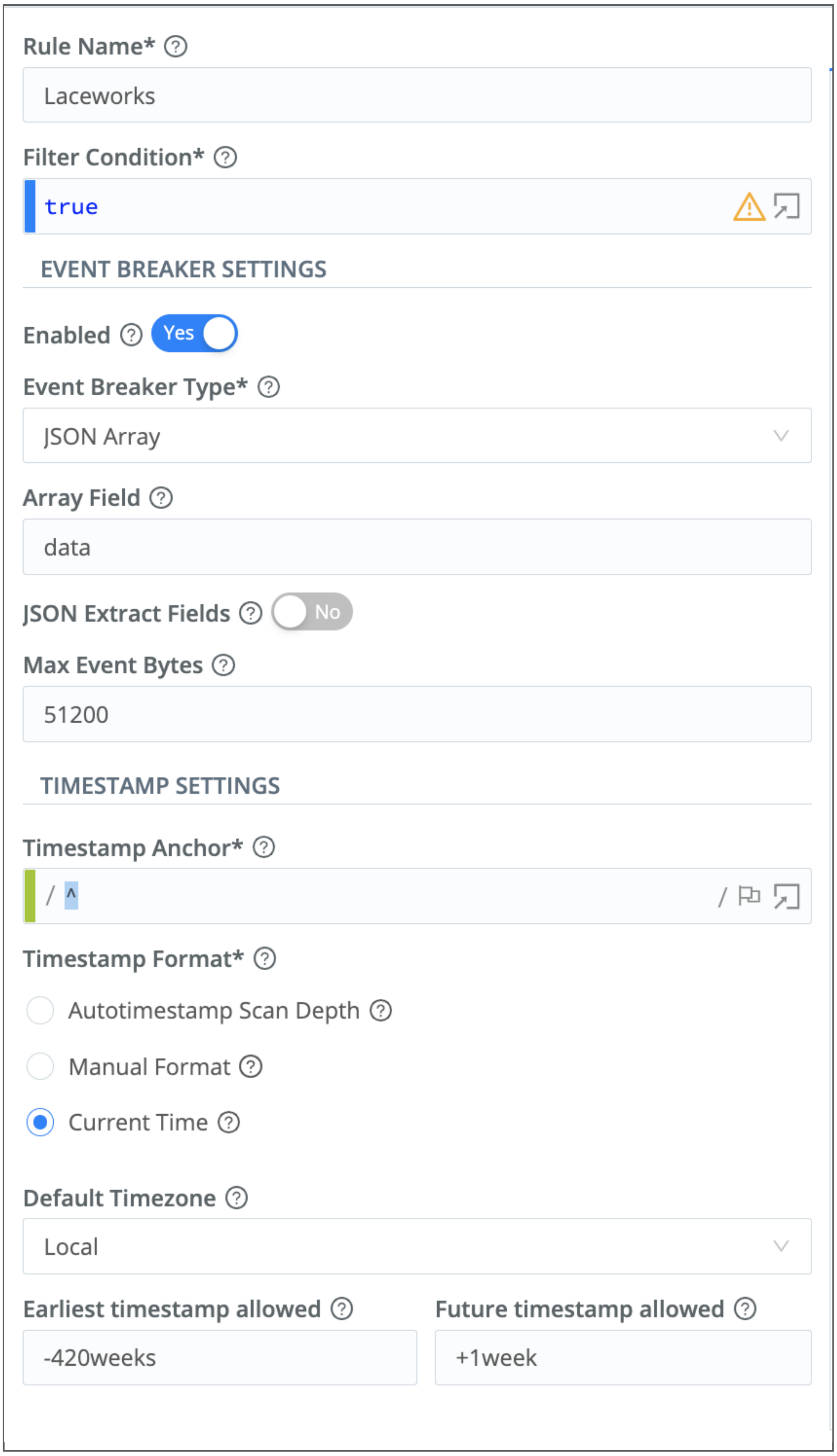
1. From Cribl Stream's top nav, select **Processing** > **Knowledge**.
2. Click the **Event Breaker Rules** left tab. From the resulting **Event Breaker Rulesets** form, click **New Ruleset** to open the **New Ruleset** modal.
3. Enter an **ID** for the new ruleset, and optionally add a **Description** and **Tags**.
4. Click **Add Rule**.

In the resulting **Rules** modal, name the Rule, and configure it as follows:

- **Filter Condition**: Enter `true`.
- **Enabled**: Toggle to `Yes`.
- **Event Breaker Type**: Select `JSON Array`.

- **Array Field**: Enter `data`.

Optionally, you can use the **Timestamp Settings** to return events whose timestamp (`_time`) matches the **startTime** or **endTime** field defined in the Lacework data.



Rule Name* ⓘ
Laceworks

Filter Condition* ⓘ
`true`

EVENT BREAKER SETTINGS

Enabled ⓘ Yes

Event Breaker Type* ⓘ
JSON Array

Array Field ⓘ
data

JSON Extract Fields ⓘ No

Max Event Bytes ⓘ
51200

TIMESTAMP SETTINGS

Timestamp Anchor* ⓘ
/ ^ /

Timestamp Format* ⓘ
○ Autotimestamp Scan Depth ⓘ
○ Manual Format ⓘ
[✓ Current Time] ⓘ

Default Timezone ⓘ
Local

Earliest timestamp allowed ⓘ
-420weeks

Future timestamp allowed ⓘ
+1week

Click **OK** to return to the previous **New Ruleset** modal, then click **Save**.

Then return to your REST Collector's configuration modal, and:

1. Select the **Result Settings** > **Event Breakers** left tab.
2. Under **Event Breaker rulesets**, click * Add ruleset**.
3. From the drop-down that now appears above the **System Default Rule**, select your new ruleset.
4. Click **Save**.

Discovering and Collecting

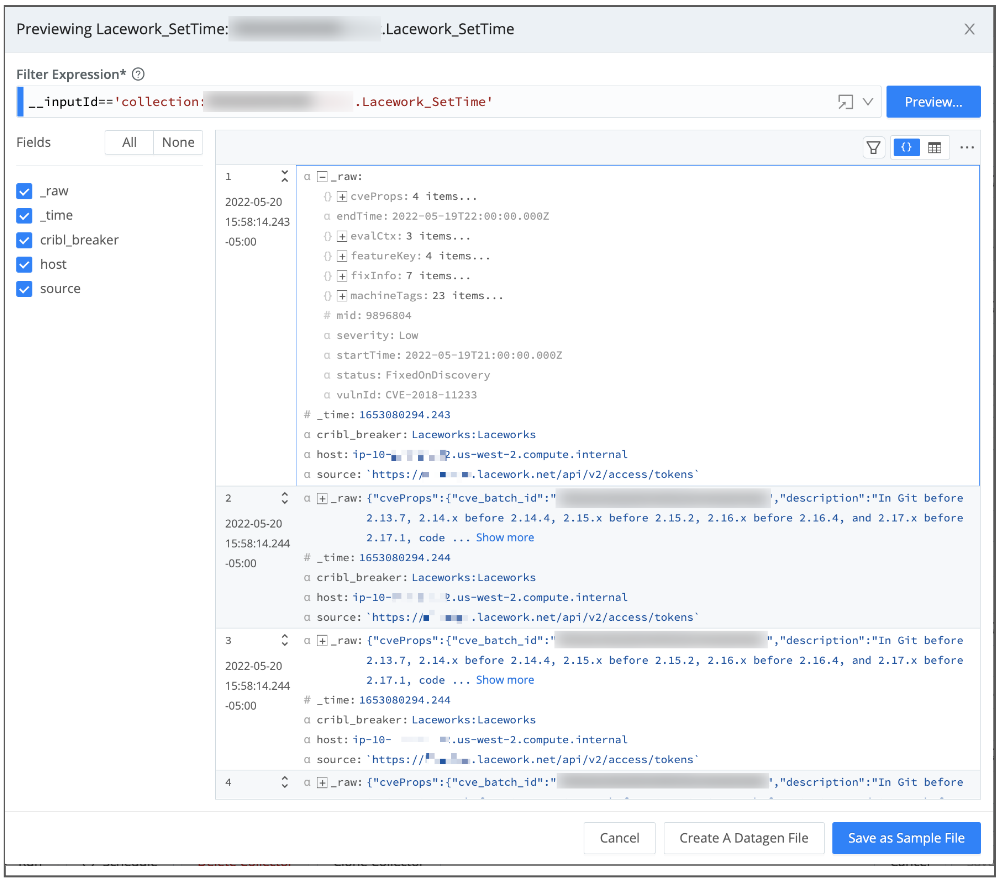
We'll start with the Discovery run:

1. On the **Manage REST Collectors** page, click **Run** beside your new Collector.
2. In the resulting **Run Collector** modal, select **Mode** > **Discovery**.
3. Configure a **Time range**, if desired.
4. Click **Run** to retrieve Discovery results.

After inspecting these results, launch the Collector run:

1. Back on the **Manage REST Collectors** page, again click **Run** beside your new Collector.
2. In the resulting **Run Collector** modal, this time select **Mode** > **Full Run**.
3. Configure a **Time range**, if desired.
4. Click **Run**.

Once the Lacework API responds, you should see data similar to this:



```
__inputId=='collection: [redacted] .Lacework_SetTime'
```

Fields: _raw, _time, cribl_breaker, host, source

```
1    2022-05-20 15:58:14.243 -05:00
  _raw:
    cveProps: 4 items...
    endTime: 2022-05-19T22:00:00.000Z
    evalCtx: 3 items...
    featureKey: 4 items...
    fixInfo: 7 items...
    machineTags: 23 items...
    mid: 9896804
    severity: Low
    startTime: 2022-05-19T21:00:00.000Z
    status: FixedOnDiscovery
    vulnId: CVE-2018-11233
  _time: 1653080294.243
  cribl_breaker: Laceworks:Laceworks
  host: ip-10-[redacted]2.us-west-2.compute.internal
  source: `https://[redacted].lacework.net/api/v2/access/tokens`
2    2022-05-20 15:58:14.244 -05:00
  _raw: {"cveProps":{"cve_batch_id":"[redacted]","description":"In Git before
        2.13.7, 2.14.x before 2.14.4, 2.15.x before 2.15.2, 2.16.x before 2.16.4, and 2.17.x before
        2.17.1, code ... Show more
  _time: 1653080294.244
  cribl_breaker: Laceworks:Laceworks
  host: ip-10-[redacted]2.us-west-2.compute.internal
  source: `https://[redacted].lacework.net/api/v2/access/tokens`
3    2022-05-20 15:58:14.244 -05:00
  _raw: {"cveProps":{"cve_batch_id":"[redacted]","description":"In Git before
        2.13.7, 2.14.x before 2.14.4, 2.15.x before 2.15.2, 2.16.x before 2.16.4, and 2.17.x before
        2.17.1, code ... Show more
  _time: 1653080294.244
  cribl_breaker: Laceworks:Laceworks
  host: ip-10-[redacted].us-west-2.compute.internal
  source: `https://[redacted].lacework.net/api/v2/access/tokens`
4
  _raw: {"cveProps":{"cve_batch_id":"[redacted]","description":"In Git before
```

Data from the Lacework API

17.5.4. Microsoft Graph API Collection

The Microsoft Graph API provides access to data in the Microsoft Cloud. This page explains how to configure a Cribl Stream REST/API Collector to ingest data using the Microsoft Graph API.

Before you start, you'll need to do the following in the Azure portal:

- Register the app you'll use to interact with Graph API.
- Generate a **Client Secret** for the app.
- Write down the **Application ID**, **Tenant ID**, and **Secret Value** defined for the app.
- Decide which API calls you expect to use, then, for each of those API calls:
    - Configure the corresponding API permissions for your app.
    - Examine how the Graph API structures events, and decide whether you want to change that structure using an Event Breaker.

For example, if you plan to call the Audit Logs API (as demonstrated in this walkthrough), do the following: <!– anything rebranded in the URL above? ->

- Configure your app with `AuditLog.Read.All` API permissions, which Audit Logs API calls require.
- Notice that the API structures Microsoft Entra ID Audit Logs data as a single JSON object with events nested under the `value` field. With an Event Breaker, you can split this into individual events for each AD Audit Log activity.

How the REST Collector Interacts with Microsoft APIs

To retrieve data using the Microsoft Graph API, your Collector first obtains a Bearer token by sending an HTTP `POST` request to the Microsoft identity platform. Once it has the Bearer token, your Collector can send an HTTP `GET` request to the Graph API, which responds with the data you requested. Configuring your Collector will be more intuitive if you keep this pattern in mind.

Configuring a Graph API REST Collector

From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Manage Sources** page's tiles or left nav. Click **New Collector** to open the **REST** > **New Collector** modal, which provides the following options and fields.

The sections described below are spread across several tabs. Click the tab links to navigate among tabs. Click **Save** when you've configured your Collector.

Collector Settings

These settings determine how data is collected before processing.

Collector ID: Unique ID for this Collector. Such as `ms_graph_42`.

Collect Settings

Collect URL: An expression that produces a URL for the collect operation. The URL can be any of the Graph API endpoints. In this example, we want to retrieve Microsoft Entra ID Audit logs, so we use the Audit Logs endpoint:

`` `https://graph.microsoft.com/v1.0/auditLogs/directoryAudits` ``

Collect method: `GET`.

Collect headers: Add the following header.

- **Name**: `content-type`.
- **Value**: `application/x-www-form-urlencoded`

To authenticate against the Microsoft Graph API, the REST Collector uses a POST call whose body is a template. The `content-type` header tells the Graph API that the template is a URL-encoded (not JSON-encoded) form. See **Authentication Settings** > **POST Body** below.

Authentication Settings

Use the **Authentication method** buttons to select **Login**, then enter the following information:

- **Username**: The **Application (Client) ID** listed on your app's **Overview** page in Azure.
- **Password**: The **Secret Value** listed on your app's **Certificate & Secret** page in Azure.
- **Login URL**: The token API endpoint for the Microsoft identity platform. Use the string: `` `https://login.microsoftonline.com/<tenant_id>/oauth2/v2.0/token` ``, substituting your Microsoft Entra ID tenant ID for `<tenant_id>`.
- A **POST body** template for the token request that the Collector sends to the Microsoft identity platform. Use the string: `` `client_secret=${password}&scope=https://graph.microsoft.com/.default&client_id=$ ``

When the Collector sends the request, it will populate the template with the **Username** and **Password** that you entered above.

- The **Token attribute** is the field that contains the Bearer token, within the Microsoft identity platform's response to the token request. Use `access_token`.
- The **Authorize expression** is a JavaScript expression that computes the `Authorization` header the Collector uses in calls to the Graph API. Use `` `Bearer ${token}` ``. The Collector will populate the `${token}` variable with the token obtained from the Microsoft identity platform.

Result Settings

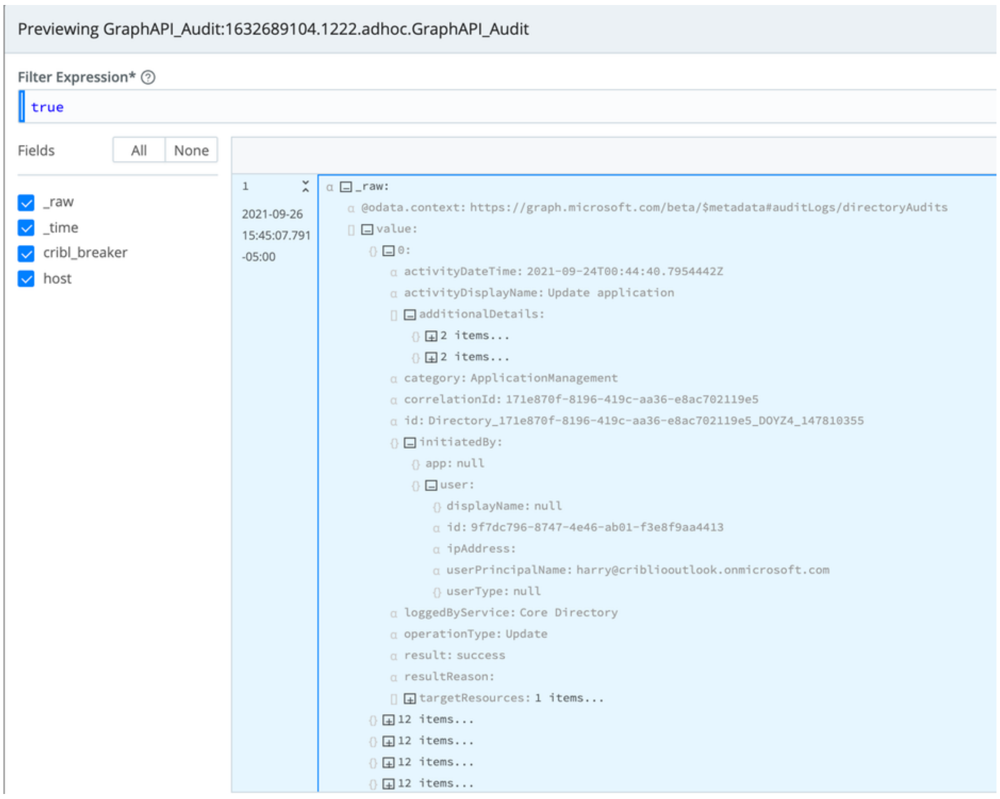
As explained above, the Microsoft Graph API delivers Microsoft Entra ID Audit Logs data as a single JSON object, with events nested under the `value` field. Cribl recommends using an Event Breaker to split this into separate name/value pairs.

You'll create the Event Breaker in the **Result Settings** > **Event Breakers** tab. But first, you must save a sample file, and create a ruleset for the Event Breaker to use.

Saving a Sample File

On the **Manage REST Collectors** page, click **Run** beside the REST Collector you configured. This opens the **Run configuration** modal, in **Preview** mode. Click **Run** to open the **Capture Sample Data** modal.

When data appears, notice that it's a large JSON object with multiple events nested under the `value` field:


Previewing GraphAPI_Audit:1632689104.1222.adhoc.GraphAPI_Audit
Filter Expression*
Fields
All
None
_raw
_time
cribl_breaker
host

Filter Expression*: true

```
1    2021-09-26 15:45:07.791 -05:00
_raw:
  @odata.context: https://graph.microsoft.com/beta/$metadata#auditLogs/directoryAudits
  value:
    0:
      activityDateTime: 2021-09-24T00:44:40.7954442Z
      activityDisplayName: Update application
      additionalDetails:
        2 items...
        2 items...
      category: ApplicationManagement
      correlationId: 171e870f-8196-419c-aa36-e8ac702119e5
      id: Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810355
      initiatedBy:
        app: null
        user:
          displayName: null
          id: 9f7dc796-8747-4e46-ab01-f3e8f9aa4413
          ipAddress:
          userPrincipalName: harry@criblioutlook.onmicrosoft.com
          userType: null
      loggedByService: Core Directory
      operationType: Update
      result: success
      resultReason:
      targetResources: 1 items...
    12 items...
    12 items...
    12 items...
    12 items...
```

Audit Logs data in original form

Click **Save as Sample File** and note the file name and location.

Creating an Event Breaker Ruleset

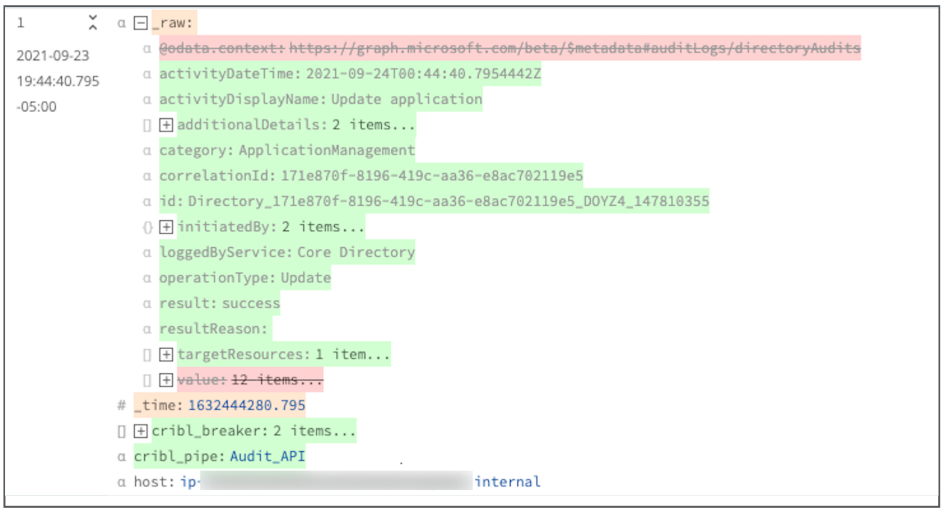
From the top menu, open **Processing** > **Knowledge** > **Event Breaker Rules**, then click **New Ruleset** to open the **New Ruleset** modal. Enter an ID and a Description for the ruleset, then click **Rules** > **Add Rule** to open the **Rules** modal.

Here's what we want the ruleset to do:

1. Remove the leading `@odata.context` statement. For this example, we'll assume that OData is not important for your use case.
2. Structure each of the fields nested within the `value` element as a separate event, where the value of the `activityDateTime` field becomes the value of the `_timestamp` field.

Name the ruleset, then:

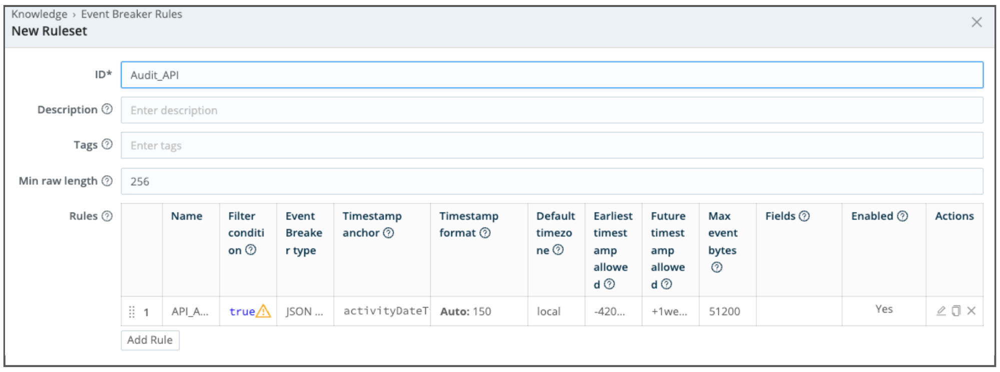
- From the **Event Breaker Type** drop-down, choose **JSON Array**. This detects that the `value` element is a JSON array, and breaks it into key-value pairs. It also discards the `@odata.context` element, because that falls outside the detected JSON array.
- In the **Timestamp Anchor** field, enter `activityDateTime` between the slashes.
- Upload your sample file.
- Compare the **In** and **Out** panes. You should see output similar to this:



```
1                 _raw:
2021-09-23          ~~@odata.context: https://graph.microsoft.com/beta/$metadata#auditLogs/directoryAudits~~
19:44:40.795        activityDateTime: 2021-09-24T00:44:40.7954442Z
-05:00              activityDisplayName: Update application
                    additionalDetails: 2 items...
                    category: ApplicationManagement
                    correlationId: 171e870f-8196-419c-aa36-e8ac702119e5
                    id: Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810355
                    initiatedBy: 2 items...
                    loggedByService: Core Directory
                    operationType: Update
                    result: success
                    resultReason:
                    targetResources: 1 item...
                    ~~value: 12 items...~~
                  _time: 1632444280.795
                  cribl_breaker: 2 items...
                  cribl_pipe: Audit_API
                  host: ip-[redacted]internal
```

Audit Logs data broken into individual events

- Click **OK** to return to the **New Ruleset** modal.
- You should see a result similar to the screenshot below. Click **Save**.



Knowledge › Event Breaker Rules
New Ruleset

ID*	Audit_API
Description	Enter description
Tags	Enter tags
Min raw length	256

Rules

	Name	Filter condition	Event Breaker type	Timestamp anchor	Timestamp format	Default timezone	Earliest timestamp allowed	Future timestamp allowed	Max event bytes	Fields	Enabled	Actions
1	API_A...	true	JSON ...	activityDateT	Auto: 150	local	-420...	+1we...	51200		Yes	

Add Rule

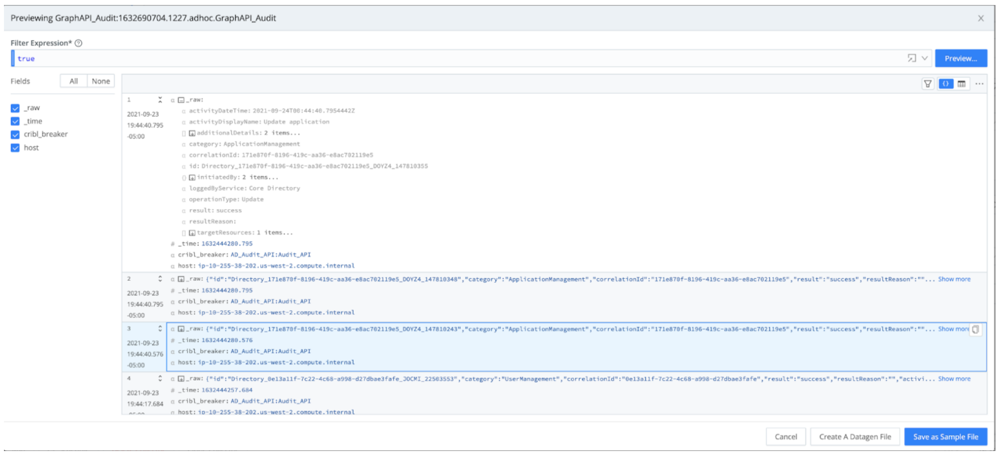
Creating an Event Breaker ruleset for Audit Logs data

Event Breakers

In the Collector's **Result Settings** > **Event Breakers** tab:

- From the **Event Breakers** drop-down, select the ruleset that you created and saved.
- Click **Save**.

Run the Collector again. Now you should see an individual JSON-formatted event for each AD Audit Log activity.



Previewing GraphAPI_Audit:1632690704.1227.adhoc.GraphAPI_Audit

Filter Expression*: true

Fields: _raw, _time, cribl_breaker, host

```
1   2021-09-23 19:44:40.795 -05:00
    _raw:
        activityDateTime: 2021-09-24T00:44:40.7954442Z
        activityDisplayName: Update application
        additionalDetails: 2 items...
        category: ApplicationManagement
        correlationId: 171e870f-8196-419c-aa36-e8ac702119e5
        id: Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810355
        initiatedBy: 2 items...
        loggedByService: Core Directory
        operationType: Update
        result: success
        resultReason:
        targetResources: 1 items...
    _time: 1632444280.795
    cribl_breaker: AD_Audit_API:Audit_API
    host: ip-10-255-38-202.us-west-2.compute.internal
2   2021-09-23 19:44:40.795 -05:00
    _raw: {"id":"Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810348","category":"ApplicationManagement","correlationId":"171e870f-8196-419c-aa36-e8ac702119e5","result":"success","resultReason":""... Show more
    _time: 1632444280.795
    cribl_breaker: AD_Audit_API:Audit_API
    host: ip-10-255-38-202.us-west-2.compute.internal
3   2021-09-23 19:44:40.576 -05:00
    _raw: {"id":"Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810243","category":"ApplicationManagement","correlationId":"171e870f-8196-419c-aa36-e8ac702119e5","result":"success","resultReason":""... Show more
    _time: 1632444280.576
    cribl_breaker: AD_Audit_API:Audit_API
    host: ip-10-255-38-202.us-west-2.compute.internal
4   2021-09-23 19:44:17.684
    _raw: {"id":"Directory_0e13a11f-7c22-4c68-a998-d27dbae3fafe_JOCMI_22503553","category":"UserManagement","correlationId":"0e13a11f-7c22-4c68-a998-d27dbae3fafe","result":"success","resultReason":"","activi... Show more
    _time: 1632444257.684
    cribl_breaker: AD_Audit_API:Audit_API
    host: ip-10-255-38-202.us-west-2.compute.internal
```

Audit Logs data from the finished Collector

17.5.5. ServiceNow API Collection

This topic covers how to configure Cribl Stream REST Collectors to gather data via ServiceNow (SNOW) REST APIs and then enrich the data using Pipelines and the Redis Function.

From among the 100-or-so SNOW REST APIs, we've chosen the CMDB Instance for this tutorial. You can adapt this material for use with other SNOW REST APIs. You'll need to create a separate Collector for each SNOW API you connect to.

The prerequisites for this setup are a ServiceNow instance, which you can obtain here, and the URL for the Redis store you wish to use.

Using the Discover and Collect Pattern

A common pattern in REST APIs is to expose two related endpoints:

- One endpoint takes a category identifier as a URL parameter, and returns a list of instances of the category, with an individual identifier for each item in the list.
- A second endpoint takes the instance identifier as a URL parameter, and returns full details about the instance.

The SNOW CMDB Instance REST API follows this pattern. The API is called "CMDB" because it exposes a Configuration Management Database which contains records that describe how pieces of hardware are configured. The categories for different kinds of hardware are specified by "classnames," and the description of an individual piece of hardware is called a Configuration Item (CI) record. The relevant API calls look like this:

- The `GET /now/cmdb/instance/{classname}` call takes a `classname` (e.g., `cmdb_ci_appl`, `cmdb_ci_linux_server`, `cmdb_ci_apache_web_server`) as a URL parameter, and returns a list of instances of the classname. Each instance has a `sys_id` and a `name`.
- The `GET /now/cmdb/instance/{classname}/{sys_id}` call takes a `sys_id` as a URL parameter, and returns the detailed CI record for the instance that the `sys_id` refers to.

In this tutorial, you'll create a Collector that uses the first API call to **discover** the systems for a given classname. Then you'll iterate the second API call over the list of `sys_id`s we discovered, to **collect** the CI records of interest. This **discover and collect** pattern works for any Cribl Stream Collector.

You'll have the Collector store the CI data in Redis. Then whatever Source you choose that has events to enrich, can do that by pulling the collected CI record data from Redis. That's the goal: enriching data.

How might this be useful? Here's one scenario: Suppose you have log data in which server names appear. Assuming that the servers in question have records in your CMDB, you can enrich the events with data about the server, such as its `sys_id` or the name of the user who updated it last.

Preparing Pipelines

Before creating the SNOW Collector itself, you'll set up the two Pipelines that the Collector needs. Both pipelines use the Cribl Stream Redis Function.

- The **process** pipeline attaches to the Collector and performs a `set` to add events to Redis.
- The **enrichment** pipeline performs a Redis `get` to retrieve the values with which we'll enrich events in the Collector.

Create the Process Pipeline

Create a pipeline called `SNOW_Instance_Process` with the following Functions and values:

Parser Function

- **Operation Mode**: `Extract`
- **Type**: `JSON Object`
- **Source Field**: `_raw`

Note that each event this Function will parse is the reponse body from a `GET /now/cmdb/instance/{classname}/{sys_id}` call—i.e., a `result` object. That object has an attribute called `attributes` whose value is itself an object containing dozens of key-value pairs. To enrich our data, we'll treat one of those (`name`, meaning server name) as a key whose value will be two more (`sys_id` and `sys_updated_by`), comma-separated. We'll specify all three as **Evaluate Fields** in the next Function.

Eval Function

Evaluate Fields:

Name	Value Expression
`name`	`result.attributes.name`
`sys_id`	`result.attributes.sys_id`
`sys_updated_by`	`result.attributes.sys_updated_by`

Redis Function

The Redis function stores our fields in the key/value relationship described earlier.

- **Command**: `set`.
- **Key**: `` `${name}` ``.
- **Args**: `` `${sys_updated_by},${sys_id}` ``.
- **Redis URL**: The URL for your Redis store, e.g., `redis://10.0.0.1:6379`.

Create the Enrichment Pipeline

Create a pipeline called `SNOW_Instance_Enrich` with the following Functions and values:

Redis Function

- **Command**: `get`.
- **Result field**: `Args`.
- **Key**: `` `${name}` ``.
- **Args**: Leave blank.
- **Redis URL**: The URL for your Redis store, e.g., `redis://10.0.0.1:6379`.

Parser Function

- **Operation Mode**: `Extract`.
- **Type**: `CSV`.
- **Source Field**: `Args`.
- **List of Fields**: `sys_updated_by sys_id`.

Eval Function

- **Remove Fields**: `Args`.

Configuring a SNOW Collector

From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Manage Sources** page's tiles or left nav. Click **New Collector** to open the **REST** > **New Collector** modal, which provides the following options and fields.

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 Collector.

Collector Settings

These settings determine how data is discovered and collected before processing. You'll see the power of the **discover and collect** pattern here: from the results of a single Discover API call, we will generate an entire set of Collect API calls. There is more about this pattern in the REST Collector documentation.

Collector ID: Unique ID for this Collector. E.g., `snow_42-a`.

Discover Settings

Discover Type: `HTTP Request`

Discover URL: An expression that produces a URL for the discover operation. We can enter this URL as a constant, e.g.:

```
'https://dev111111.service-now.com/api/now/cmdb/instance/cmdb_ci_appl'
```

Discover method: `GET`

Discover Data Field: `result`

We specify `result` for the **Discover Data Field** because that is the key of the JSON array that our Discover call retrieves. For example:

```
  "result": [
    {
      "sys_id": "3a290cc60a0a0bb400000bdb386af1cf",
      "name": "PS LinuxApp01"
    },
    {
      "sys_id": "3a5dd3dbc0a8ce0100655f1ec66ed42c",
      "name": "PS LinuxApp02"
    }
  ]
```

This array becomes the list of items for which Collect tasks will be created. We'll reference one of its attributes, namely `sys_id`, as a variable in the Collect task's URL.

Collect Settings

Collect URL: An expression that produces a URL for the collect operation. We enter the first part of the URL as a constant, then for the final URL parameter, we use an expression to reference the `sys_id` returned by the Discover call:

```
'https://dev111111.service-now.com/api/now/cmdb/instance/cmdb_ci_appl'+ C.Encode.ui
```

We use the C.Encode.uri Cribl expression to encode the `sys_id` in case any `sys_id` contains unsafe characters.

**Collect method **:`GET`.

Authentication

Use the **Authentication method** drop-down to select one of these options:

- **None**: Don't use authentication.
- **Basic**: Use HTTP token authentication.
- **Basic (credentials secret)**: Select a stored text secret in the resulting drop-down, or click **Create** to configure a new secret.
- **Login**: This option requires you to provide:
  - The **Username** and **Password** fields for your HTTP Basic authentication credentials.

  - The **Login URL** that SNOW should use to connect to Cribl Stream.
  - A **POST Body** template for the request SNOW uses when logging in. You must edit this if your credentials' location differs from that specified by default.
  - The **Token Attribute**, which is the path to the token attribute in login response body. Nested attributes are OK.
  - The **Authorize Expression**, a JavaScript expression that computes the `Authorization` header to pass in Discover and Collect calls. The value `${token}` references the token obtained from login.
- **Login (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. You must also provide the **Login URL**, **POST Body**, **Token Attribute**, and **Authorize Expression**, as in the plain **Login** option.

Result Settings

These settings enable the Collector you've been configuring to use your **process** Pipeline.

Result Routing

Toggle **Send to Routes** to `No`.

Choose the `SNOW_Instance_Process` pipeline from the dropdown.

Choose whatever **Destination** suits your purposes; a DevNull Destination would make sense given that the purpose of this Collector is simply to get data into Redis.

Adding a Source with Data to Enrich

At this point, you have a Collector which stores CI record data in Redis. Assuming that you have a Source with events you want to enrich, you should now configure that Source to use the **enrich** Pipeline you created earlier.

In **Processing Settings** > **Pre-Processing**, choose `SNOW_Instance_Enrich` from the **Pipeline** drop-down.

Verifying the Enrichment of Events

This procedure should demonstrate that your setup is working, or help you troubleshoot.

1. Send a request to the SNOW Instance API to verify that it returns data.

    - Try the query builder.

2. Run the Collector once. Since you have attached the **process** Pipeline to the Collector, it should update Redis with `set` calls.
    - When you're ready to move towards production, you can run the Collector on a schedule, to update Redis periodically.
3. Run a `get` command on the Redis command line, using the key for your chosen API, to verify that Redis was updated as expected.
4. Check the Source whose data you are enriching, to verify that the fields you added are showing up.

17.5.6. Wiz API Collection

This topic covers how to configure Cribl Stream REST Collectors and Event Breakers to gather data from the Wiz cloud security platform. The REST Collectors will communicate with APIs that your organization's Wiz portal exposes. You'll create and configure four REST Collectors, one for each of these Wiz APIs: Audit Logs, Configuration Findings (sometimes called Cloud Configuration), Issues, and Vulnerabilities.

Instead of configuring REST Collectors setting-by-setting, you will first download them from Cribl's Collector Templates repository and then import them into Cribl Stream using the **Manage as JSON** feature, replacing placeholders in the configuration with values from your Wiz portal. This approach is faster and less error-prone than clicking your way through the relevant UIs, especially since the configurations required to interact with the Wiz APIs are quite complicated.

Prerequisites From Wiz

This topic assumes that when you became a Wiz customer, Wiz sent the following items:

Item	Value
Client ID	53-character string
Client secret	64-character string
Auth URL	`https://auth.app.wiz.io/oauth/token` or similar
API endpoint	`https://api.us17.app.wiz.io/graphql` or similar

These items should be the same across all Wiz APIs. The Wiz platform differentiates requests to one API from requests to another by inspecting the request body. (The configurations supplied later in this topic include an appropriate request body for each API.)

Note the values that Wiz sent you, and keep them handy for when you configure the REST Collectors later on.

Downloading the Collectors

Click the links to open the following files from Cribl's Collector Templates repository, and download the files to a convenient location on your local system.

1. Audit Logs Collector: `collector-wiz-auditlogentries.json`
2. Configuration Findings Collector: `collector-wiz-configurationfindings.json`

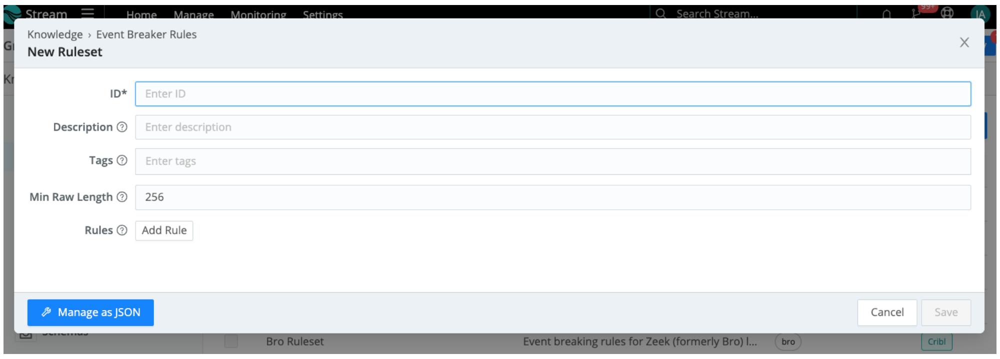
3. Issues Collector: `collector-wiz-issues.json`
4. Vulnerabilities Collector: `collector-wiz-vulnerabilities.json`

Configuring the Event Breaker

Although you will create four REST Collectors, you only need to add a single Event Breaker whose ruleset contains four rules – one for each Wiz API – enabling it to break events for all four REST Collectors.

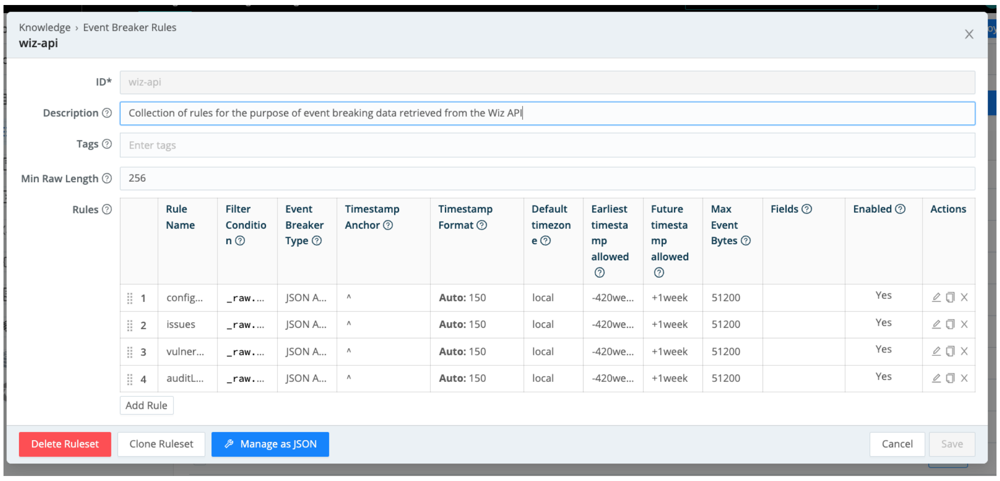
The Event Breaker strips headers from events, leaving only the records of interest; and, it captures timestamps in the proper way.

1. Navigate to **Manage** > **Processing** > **Knowledge** > **Event Breaker Rules**.
2. Click **Add Ruleset**.
3. Click **Manage as JSON** at lower left to open the JSON editor.
4. Click this link to open the Event Breaker file (`breaker.json`) in the Collector Templates repository, then click the **Copy icon** to copy the Event Breaker.
5. Back in the Cribl Stream UI, paste the Event Breaker into the editor (replacing the default object that the editor opens with).
6. Click **Save**.



The Event Breaker UI before you add the ruleset

When you have finished adding the Event Breaker, you should see that it contains four rules:


Knowledge › Event Breaker Rules
wiz-api

ID*: wiz-api
Description: Collection of rules for the purpose of event breaking data retrieved from the Wiz API
Tags: Enter tags
Min Raw Length: 256

Rules:

	Rule Name	Filter Condition	Event Breaker Type	Timestamp Anchor	Timestamp Format	Default timezone	Earliest timestamp allowed	Future timestamp allowed	Max Event Bytes	Fields	Enabled	Actions
1	config...	`_raw.…`	JSON A...	^	**Auto:** 150	local	-420we...	+1week	51200		Yes	
2	issues	`_raw.…`	JSON A...	^	**Auto:** 150	local	-420we...	+1week	51200		Yes	
3	vulner...	`_raw.…`	JSON A...	^	**Auto:** 150	local	-420we...	+1week	51200		Yes	
4	auditL...	`_raw.…`	JSON A...	^	**Auto:** 150	local	-420we...	+1week	51200		Yes	

The complete Event Breaker ruleset

Configuring the REST Collectors

You'll need to complete the procedure in this section four times. Each time, you'll create one REST Collector to handle a particular Wiz API.

To learn more about REST Collectors, see our REST/API Endpoint and Scheduling and Running topics.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Data Sources** page's tiles or the **Sources** left nav. Click **Add Collector** to open the **REST** > **New Collector** modal.

1. Click **Manage as JSON** to open the configuration editor.
2. Click **Import** and select the desired Collector from among the four you downloaded above:
   - Audit Logs Collector (`collector-wiz-auditlogentries.json`)
   - Configuration Findings Collector (`collector-wiz-configurationfindings.json`)
   - Issues Collector (`collector-wiz-issues.json`)
   - Vulnerabilities Collector (`collector-wiz-vulnerabilities.json`)
3. In the **Replace Placeholder Values** modal, enter the values you noted earlier into the appropriate fields:
   - For `loginUrl`, enter the value of **Auth URL**.
   - For `clientSecretParamValue`, enter the value of **Client secret**.
   - In the `authRequestParams` array, find the object where `name` has the value `client_id`, and replace the value of `value` with the value of **Client ID**.
   - For `collectUrl`, enter the value of **API endpoint**.

4. Click **Replace Values** to close the modal.
5. Click **OK** to finish configuring the Collector.

Once you've created all four REST Collectors, you'll be ready to start collecting from the Wiz APIs.

Discovering and Collecting

You can perform the procedures in this section with any of the four REST Collectors you've created. To configure time ranges, you'll use **Earliest** and **Latest** values.

- If you enter **Earliest** and **Latest** values, Cribl Stream will pass them into the API query.
- If you leave these fields blank, Cribl Stream will query the API for the 24 hours before the job is scheduled to run.

We'll start with the Discovery run:

1. On the **Manage REST Collectors** page, click **Run** beside your new Collector.
2. In the resulting **Run Collector** modal, select **Mode** > **Discovery**.
3. Configure a **Time range**, if desired.
4. Click **Run** to retrieve Discovery results.

After inspecting these results, launch the Collector run:

1. Back on the **Manage REST Collectors** page, again click **Run** beside your new Collector.
2. In the resulting **Run Collector** modal, this time select **Mode** > **Full Run**.
3. Configure a **Time range**, if desired.
4. Click **Run**.

Automating your Collector Runs

1. On the **Manage REST Collectors** page, click **Schedule** in the **Actions** column for your new Collector. The **Run Collector** modal will open.
2. Toggle **Enable** on and configure the scheduling options as desired.
3. Click **Save**.

Beware of Wiz API Results Limits

To avoid degrading the performance of your Wiz environment, follow the principles described in this section.

- Do not exceed three API requests per second. Cribl Stream is generally not used in a manner that would approach this limit but if you do, the Wiz API will likely respond with `HTTP 429 Too Many Request` errors.
- Use filters in the query string to ensure that your queries pull only the relevant information.
- If you choose to capture historical data during your initial integration, keep each API's results limits in mind when running ad-hoc API collections.
- For ongoing collection, use incremental collections. To do this, schedule the REST collector to run over a timeframe that suits the task. For example, the Wiz Vulnerabilities Report API is for incremental use only and is best scheduled to run daily, pulling results only from the previous day.

The table below shows the results limits for each Wiz API.

API	Limit (results)
Audit Logs	10,000
Cloud Configuration	10,000
Issues	no limit
Vulnerability	no limit

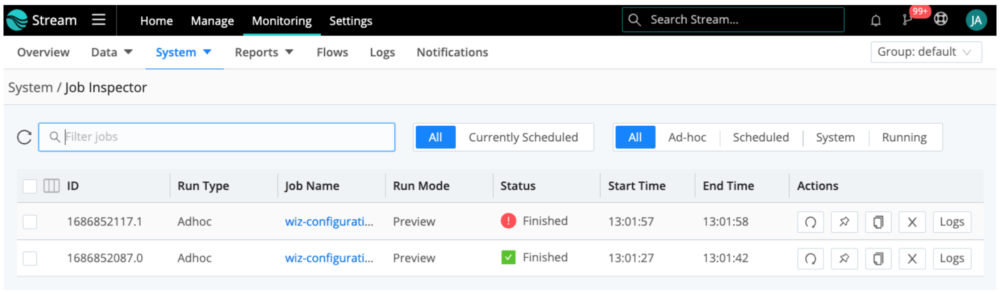
Once your query hits the limit of results for a given Wiz API, the API will send no further results.

Troubleshooting

The REST Collector treats all non-`200` responses from configured URL endpoints as errors. On Discover, Preview, and most Collect jobs, it interprets these as fatal errors. On Collect jobs, a few exceptions are treated as non-fatal:

- Where a Collect job launches multiple tasks, and only a subset of those tasks fail, Cribl Stream places the job in failed status, but treats the error as non-fatal. (Note that Cribl Stream does not retry the failed tasks.)
- Where a Collect job receives a 3xx redirection error code, it follows the error's treatment by the underlying library, and does not necessarily treat the error as fatal.

Detailed logging is available for each Collector run to help troubleshoot issues or explore the results of each collector run. To view the jobs, click Monitoring->System->Job Inspector as shown below.



ID	Run Type	Job Name	Run Mode	Status	Start Time	End Time	Actions
1686852117.1	Adhoc	wiz-configurati...	Preview	Finished	13:01:57	13:01:58	Logs
1686852087.0	Adhoc	wiz-configurati...	Preview	✓ Finished	13:01:27	13:01:42	Logs

The Job Inspector

17.5.7. Creating A Custom Collector

Cribl's flagship product introduced Collectors in Logstream 2.2, and they have since evolved to become a critical part of the platform. You can think of Collectors as jobs that retrieve data from an external service. You can also schedule them to run periodically, like a Linux cron job.

We built the Collection framework to be extensible like Cribl Stream Functions. This means that you can create new Collectors on the fly. This page covers the Collection process, schema files, and their implementation – to give you some context – before walking you though the steps of creating your own Collector.

Collection Process

At its simplest, collection is a two-step process: Discover, and Collect.

Discover

This step identifies the items to collect. The Discover call generates a list of items, and Collect tasks will be created for those items. Note that:

- The Discover call can return zero to many items. The Collection phase will run only if Discover returned at least one item.
- Your Collector's purpose (and definition) determines the actual data that Discover will return.
- If the Collector pulls data from files on disk, the Discover call lists the files in the directory to determine how many items match the criteria (matching by filter and/or date range). Assuming that there are 100 files to collect from, the Discover call will return 100 items (each specifying a file path and size).

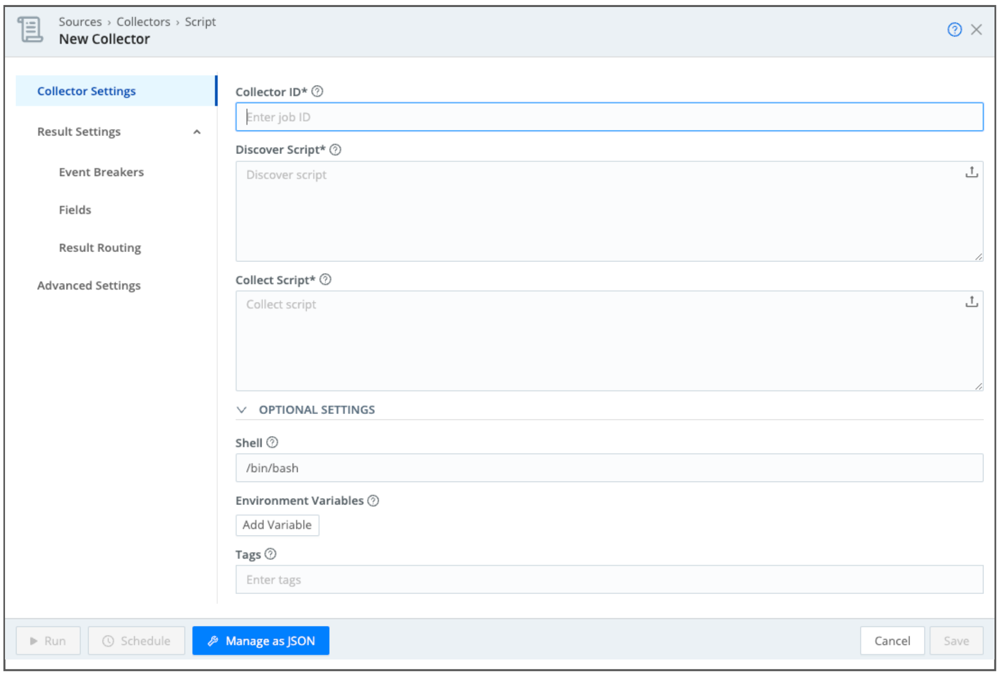
Collect

Each item, from the Discover phase's returned list of items, will trigger one Collect task for Cribl Stream to create and run. More on Collect:

- Generally, there is a single Collect task per item returned by Discover.
- In distributed deployments, you configure Collectors at the Worker Group level, and Worker Nodes execute the tasks. However, the Leader Node oversees the task distribution, and tries to maintain a fair balance across jobs.

Collector Schema Files

The Collector schema files, `conf.schema.json` and `conf.ui-schema.json`, describe the structure of the UI for configuring a new or existing Collector. As an example, here's a Script Collector's Configuration modal:



Script Collector Configuration Screen

The `conf.schema.json` below defines all the fields displayed on that **Collector Settings** form:

```
{
  "type": "object",
  "title": "",
  "required": ["discoverScript", "collectScript"],
  "properties": {
    "discoverScript": {
      "type": "string",
      "title": "Discover Script",
      "minLength": 1,
      "description": "Script to discover what to collect. Should output one task pe
    },
    "collectScript": {
      "type": "string",
      "title": "Collect Script",
      "minLength": 1,
      "description": "Script to run to perform data collections. Task passed in as
    },
    "shell": {
      "type":"string",
      "title": "Shell",
      "description": "Shell to use to execute scripts.",
      "default": "/bin/bash"
    }
  }
}
```

The `conf-ui-schema.json` file further refines the field specifications in `conf.schema.json`. In the example below, it specifies that the `discoverScript` and `collectScript` fields should use the `TextareaUpload` widget with 5 rows. This file also gives each text field some placeholder (ghost) text, to display when no data is present in the field.

```
{
  "discoverScript": {
    "ui:widget": "TextareaUpload",
    "ui:options": {
      "rows": 5
    },
    "ui:placeholder": "Discover script"
  },
  "collectScript": {
    "ui:widget": "TextareaUpload",
    "ui:options": {
      "rows": 5
    },
    "ui:placeholder": "Collect script"
  }
```

Schemas can be simple, like the Script Collector, or complex like the REST Collector. This guide doesn't cover all the possibilities of working with schemas, which do get complex. However, you can check out other existing Collectors as examples of how to apply schemas for your own use case.

Collector Implementation

The Collector's implementation logic is part of `index.js`, and resides in the same directory as the Collector schema files. You must define the following attributes and methods for the Collector:

```
// Jobs can import anything from the C object, to see what's available use the
// Stream UI and an Eval Function to discover options.
const { Expression, PartialEvalRewrite } = C.expr;
const { httpSearch, isHttp200, RestVerb, HttpError, wrapExpr, DEFAULT_TIMEOUT_SECS

exports.name = 'Weather';
exports.version = '0.1';
exports.disabled = false; // true to disable the collector
exports.destroyable = false;
exports.hidden = false; // true to hide collector in the UI

// Define Collector instance variables here: i.e.:
// let myVar; // Initialize in init, discover, or collect.

// init is called before the collection job starts. Gives the Collector a chance
// to validate configs and initialize internal state.
exports.init = async (opts) => {
  // validate configs, throw Error if a problem is found.
  // Initialize internal attributes here
}

// The Discover task's main job is to determine 'what' to collect. Each item
// to collect (i.e. a file, an API call, etc) is reported via the job object
// and will execute a collection task.
// Note that different instances of the Collector will be used for the Discover
// and Collect operations. Do not set internal Collector state in Discover
// and expect it to be present in Collect. The _init method will be called
// prior to Discover orCcollect.
exports.discover = async (job) => {
  // Job is an object that the collector can interact with, for example
  // we can use the job to access a logger.
  job.logger().info('Discover called');

  // In this case, reporting 2 hard coded items to be collected. Normally,
  // collectors dynamically report items to collect based on input from
  // an API or library call.
  await job.addResults([{"city": "San Francisco"},{"city": "Denver"}]);
  // Can also add results one at a time using await job.addResult("city": "San Fra
}

// One invocation of the Collect method is made for each item reported by the
// discover method. The collectible object contains the data reported by discover.
// In our example collect will be called twice, once for each item returned
// by discover's addResults call:
// Invocation 1: {"city": "San Francisco"}
// Invocation 2: {"city": "Denver"}
exports.collect = async (collectible, job) => {
   job.logger().info('In collect', { collectible });
   try {
     // Do actual data collection here. In this case we might make a REST API call
     // to retrieve current weather conditions for the city in collectible.
     const myReadableStream = doGetWeather(collectible.city);
   } catch (error) {
     // If the collector encounters a fatal error, pass the error to the job. This
     // will make the error visible in the Job inspector UI.
     job.reportError(error)
     return;
   }
   // Return result of the collect operation should be Promise<Readable>
   // which will be piped to routes or to the configured pipeline and destination.
   return Promise.resolve(myReadableStream)
}
```

Set Up a New Collector

Now to the exciting part. Here's an overview of how to add a Collector:

1. Create a directory where Collector files will reside. All files associated with the Collector should be in this directory.
2. Create the schema files and add them to the directory. The following schema files describe the structure of the UI for configuring a new or existing Collector:
   - `schema.json`
   - `ui-schema.json`
3. Create a JavaScript file named `index.js`, with naming attributes and required methods.
4. Test and validate the Collector.
5. Install the Collector in Cribl Stream.

Before You Start

A few things to note before we start the process of adding a custom Collector:

1. For a standalone install of a running Cribl Stream instance, add new directories and files to the `$CRIBL_HOME/local/cribl/collectors` directory (e.g., `/opt/cribl`).

2. The Collector will show up in the UI only if all schema files and the `index.js` file successfully compile. If the Collector is not showing up in the UI, check whether there was a problem compiling one of the files. You can check the errors through the API Server Logs.
3. Develop your Collector in a local test environment, as a best practice. After making changes to the Collector, you might need to restart/and or deploy your system.
4. Collectors can access all Node.js built-in modules, using the `require` directive to import each module.
5. You can include third-party Node.js modules into a custom Collector by installing them in the Collector's home directory.
6. Cribl Stream ships with out-of-the-box Collectors which reside in the `$CRIBL_HOME/default/cribl/collectors` directory. Feel free to copy contents from one of the existing Collectors to your `$CRIBL_HOME/local/cribl/collectors` (local directory) as a starting point/example of how to build more-complex schemas.

Never modify a Collector in the default directory (`$CRIBL_HOME/default/cribl/collectors/*`). Because:

1. Doing so changes the behavior of a Collector in your installed system.
2. Contents of the default directory will be overwritten during upgrades.

When creating your own Collector, always make a copy to a directory in the local `$CRIBL_HOME/local/cribl/collector/<yourCollector>`

Sample Collector Requirements

For this example, our sample Collector's goal is to generate events – containing a random quote obtained from an internal list – for a list of users. Here is a breakdown of this Collector's requirements:

1. Provide a random quote to a predefined or random list of usernames.
2. Accept the names of users for whom to generate quotes.
3. Accept randomly generated usernames, by specifying the number of usernames to generate.
4. Generate a single event containing a random quote for each user.
5. Pick the random quotes from a hard-coded list, or optionally, use a REST API instead.

For this sample Collector, we'll reference a third-party Node module to generate random usernames, and to give you a basic understanding of how to reference external code packages.

Set Up Your Environment

For this guide, we'll build a custom Collector in a standalone Cribl Stream install running in Docker. To adapt the steps for a distributed environment:

1. Build the Collector on the Leader Node.
2. The Collector directory will reside in the filesystem at: `$CRIBL_HOME/groups/<workerGroup>/local/cribl/collectors/quote_generator`. For example, to build in the default Worker Group, work in the directory: `$CRIBL_HOME/groups/default/local/cribl/collectors/quote_generator`.
3. Before running the Collector, commit and deploy changes for the parent Worker Group.

Deploy a Single Instance of Cribl Stream in Docker

1. Start the Docker instance by running the following command:
   `docker run -d -p 19000:9000 cribl/cribl:latest`
2. List Docker containers by running `docker ps`, as shown here:

```
$ docker ps
CONTAINER ID    IMAGE                     COMMAND                    CREATED
544370698fb5    cribl/cribl:3.4.1-RC1     "/sbin/entrypoint.sh…"     4 minutes ago
```

3. Access the Cribl Stream UI at port: `http://localhost:19000`
4. Connect to the container:
   `docker exec -it <Container ID> bash`
   (e.g., `docker exec -it 544370698fb5 bash`)
5. Update the apt installer:
   `apt update`
6. Install Vim (or an editor of your choice):
   - vim: `apt install vim`
   - nano: `apt install nano`

After this, remain connected to the Docker container, and follow the steps below to create the Collector.

Remember to leave the Docker container running while you work on the Collector. Also, remember to explicitly back up your files before stopping the Docker container.

Build a Collector (Single-Instance Environment)

In your terminal, type the following commands:

1. `cd /opt/cribl/local/cribl`

2. `mkdir collectors`
3. `cd collectors`
4. `mkdir quote_generator`
5. `cd quote_generator`
6. `cp ../../../../default/cribl/collectors/script/* .`
7. `chmod +w index.js *.json`
8. Edit `index.js` to change the following, to assign a unique name to the Collector:

   ```
   exports.name = 'Quote Generator';
   ```

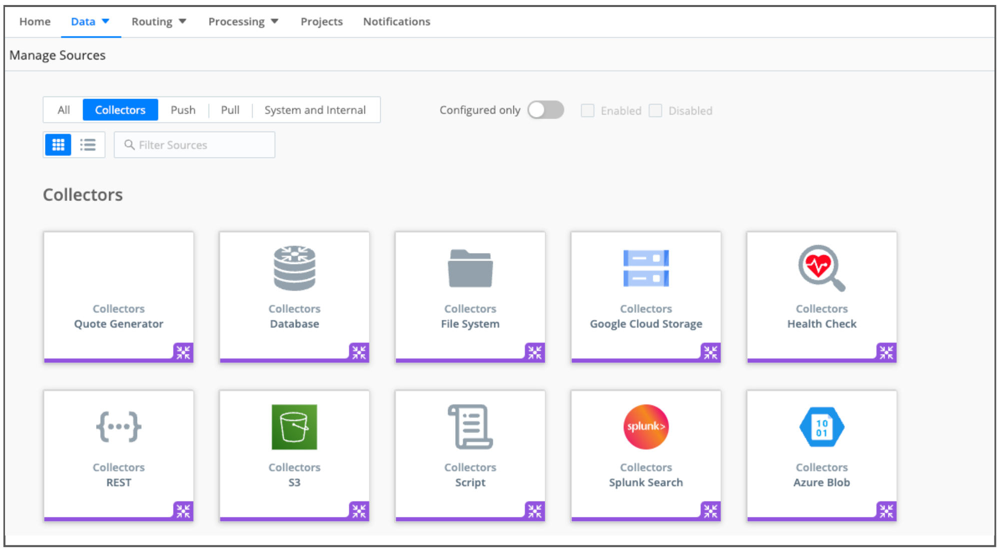
9. Edit `properties.type` within `conf.schema.json` file to remove `flag` keyword and change the `enum` to `["quote_generator"]`. You should end up with something like this:

   ```
   {
     "type": "object",
     "title": "",
     "required": ["discoverScript", "collectScript"],
     "properties": {
       "type": {
         "type": "string",
         "enum": ["quote_generator"]
       },
       (...)
   ```

10. Next, restart Cribl Stream with the following command:
    `$ /opt/cribl/bin/cribl restart`
11. After Cribl Stream restarts, you must log out and log back in for the new Collector tile to display under Sources.

Configure the Collector in Cribl Stream

1. From your Cribl Stream UI's top nav, select **Data** > **Sources**, then select **Collectors** > **Quote Generator** from the **Data Sources** page's tiles or the **Sources** left nav.

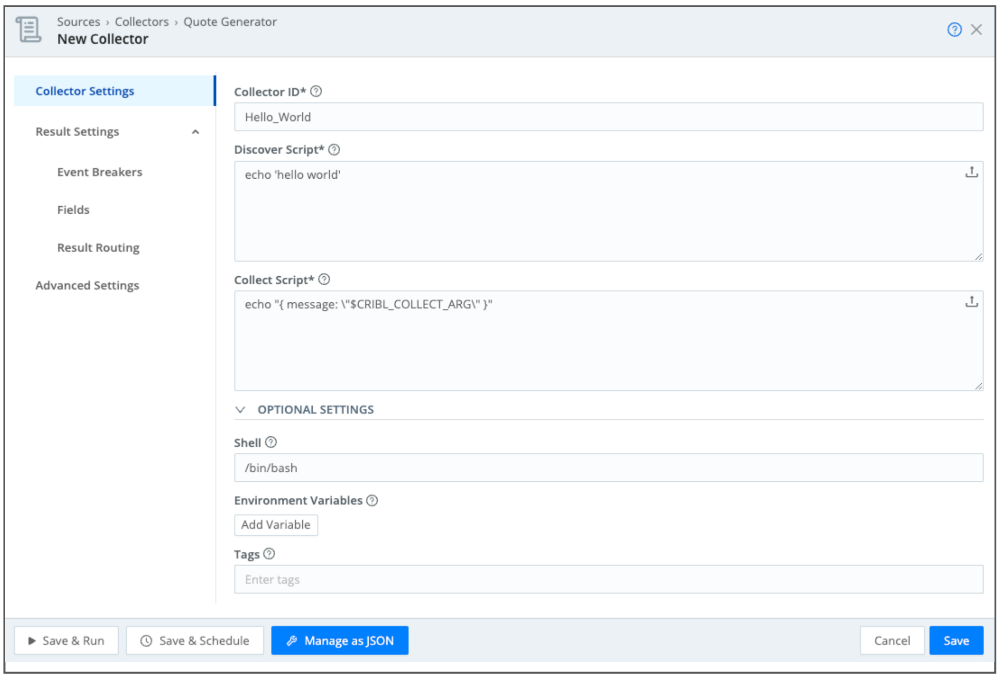


Quote Generator Collector (Missing an Icon)

2. Click **Add Collector** to open the **Quote Generator** > **New Collector** modal.

3. Enter the following in the **Collector Settings** tab, then click **Save**:

   - **Collector ID**: `Hello_World`
   - **Discover Script**: `echo "hello world"`
   - **Collect Script**: `echo "{ message: \"$CRIBL_COLLECT_ARG\" }"`



Collector ID*: Hello_World

Discover Script*: echo 'hello world'

Collect Script*: echo "{ message: \"$CRIBL_COLLECT_ARG\" }"

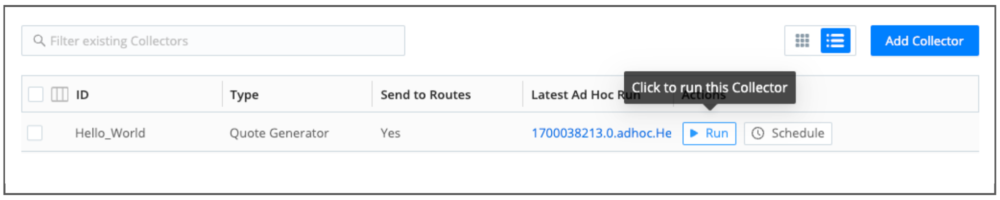
Shell: /bin/bash

Environment Variables

Tags: Enter tags

Collector Settings Tab

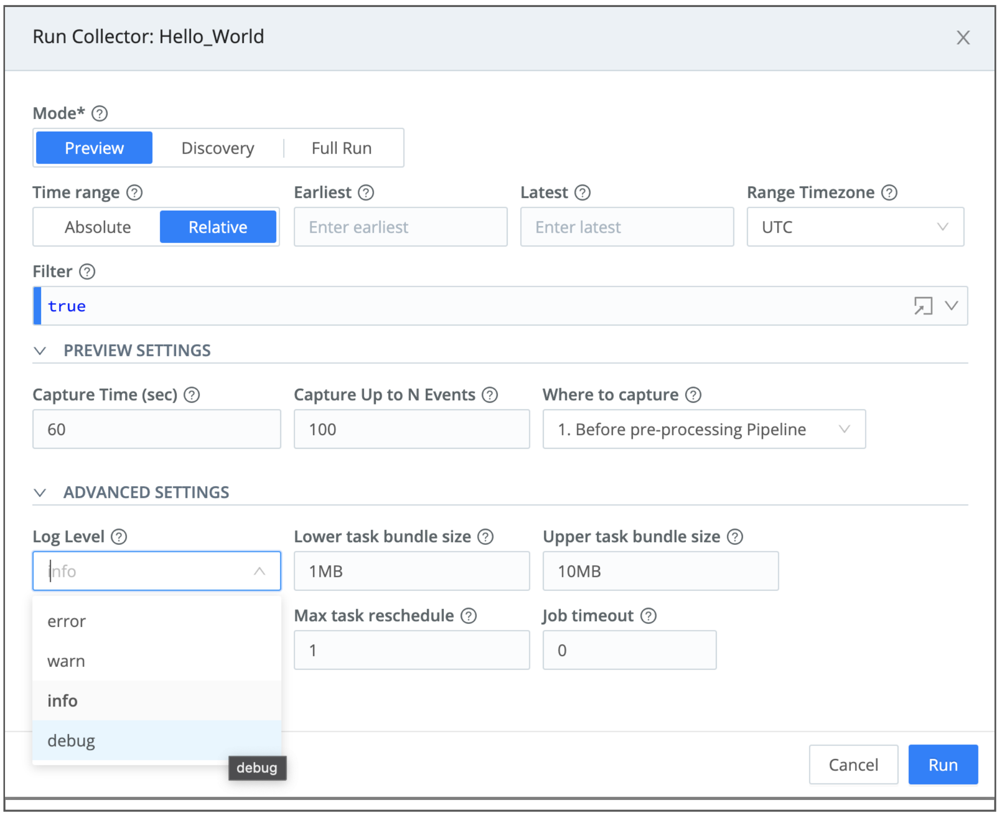
4. On the **Manage Quote Generator Collectors** page, click **Run** next to the Collector.



ID	Type	Send to Routes	Latest Ad Hoc Run
Hello_World	Quote Generator	Yes	1700038213.0.adhoc.He

Run the Collector

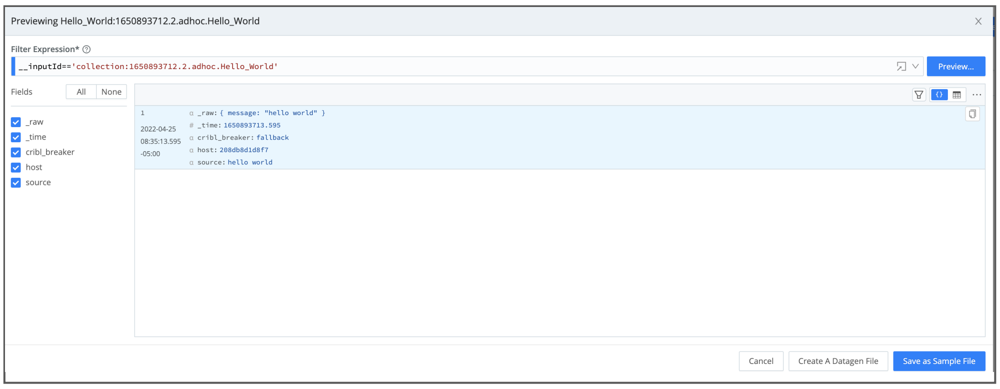
5. In the **Run configuration** modal, click **Run** again. Note that you can set the Collector's debug level in the modal's **Advanced Options** section. For details, see Run Configurations and Shared Settings.



Time range: Relative
Earliest: Enter earliest
Latest: Enter latest
Range Timezone: UTC
Filter: `true`
Capture Time (sec): 60
Capture Up to N Events: 100
Where to capture: 1. Before pre-processing Pipeline
Log Level: info
Lower task bundle size: 1MB
Upper task bundle size: 10MB
Max task reschedule: 1
Job timeout: 0

Run the Collector

6. After the Collector runs, the following event will be generated:



```
__inputId=='collection:1650893712.2.adhoc.Hello_World'
```

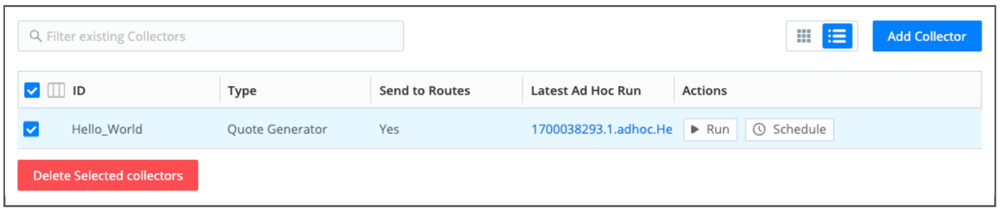
Fields: _raw, _time, cribl_breaker, host, source

```
1                  _raw: { message: "hello world" }
2022-04-25         _time: 1650893713.595
08:35:13.595       cribl_breaker: fallback
-05:00             host: 208db8d1d8f7
                   source: hello world
```

Run the Collector

Now that we have a basic shell for our Collector, let's change the UI and `index.js` to add our custom Collector logic. Before making any additional changes, delete the Collector we just created.

7. On the **Manage Quote Generator Collectors** page, click the check box next to your Collector, then click **Delete selected Collectors**.



ID	Type	Send to Routes	Latest Ad Hoc Run	Actions
Hello_World	Quote Generator	Yes	1700038293.1.adhoc.He	Run Schedule

Delete the Collector

Edit the Schema Files

1. Replace the contents of `conf.schema.json` with the following:

```
{
  "type": "object",
  "title": "",
  "required": ["autoGenerateNames"],
  "properties": {
    "autoGenerateNames": {
      "type": "boolean",
      "title": "Auto generate names",
      "description": "Turn on to autogenerate names, use num names to specify
      "default": false
    }
  },
  "dependencies": {
    "autoGenerateNames": {
      "oneOf": [
        {
          "required": ["numNames"],
          "properties": {
            "autoGenerateNames": { "enum": [true] },
            "numNames": {
              "type": "number",
              "title": "Num names",
              "minimum": 1,
              "maximum": 1000,
              "description": "The number of names to auto generate, each name
            }
          }
        },
        {
          "required": ["names"],
          "properties": {
            "autoGenerateNames": { "enum": [false] },
            "names": {
              "type": "array",
              "title": "Names",
              "minLength": 1,
              "description": "List of user names to retrieve quotes for.",
              "items": {"type": "string"}
            }
          }
        }
      ]
    }
  }
}
```

Two interesting things to note in the `conf.schema.json` file:

- When you disable – set to `false` – the boolean toggle `autoGenerateNames`, the UI displays a **Names** field requesting a list of user names for which to retrieve quotes.
- When you enable – set to `true` – the boolean toggle `autoGenerateNames`, the UI displays a **Number of names to enter** field requesting the number of names to auto-generate.

The dependency allows dynamic onscreen behavior based on the field's value.

2. Replace the contents of `conf.ui-schema.json` with the following:

```
{
  "names": {
    "ui:field": "Tags",
    "ui:placeholder": "Enter names",
    "ui:options": {
      "separator": ","
    }
  }
}
```

The UI schema, in this case, further refines the widget ("Tags") used for the Names field. We will see this in action a bit later.

3. Finally, replace the contents of `index.js` with the following:

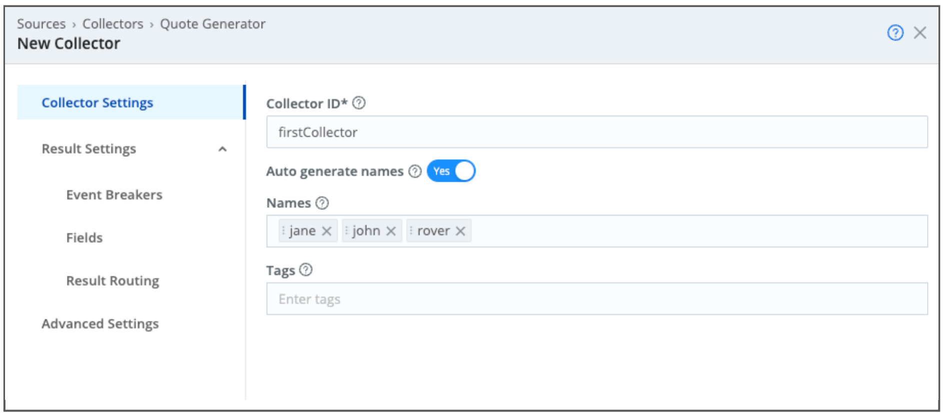
```
exports.name = 'Quote Generator';
exports.version = '0.1';
exports.disabled = false;
exports.destroyable = false;

const Readable = require('stream').Readable;
const os = require('os');
const host = os.hostname();
const logger = C.util.getLogger('myCollector'); // For use in debugging init,

let conf;
let batchSize;
let filter;
let autoGenerate = false;
let numNames = 0;
let names;

exports.init = async (opts) => {
  conf = opts.conf;
  //logger.info('INIT conf', { conf });
  batchSize = conf.maxBatchSize || 10;
  filter = conf.filter || 'true';
  autoGenerate = conf.autoGenerateNames
  names = conf.names || [];
  numNames = conf.numNames ?? 3;
  //logger.info('INIT VALUES', { autoGenerate, names, numNames });
  return Promise.resolve();
};

exports.discover = async (job) => {
  for (let i = 0; i < names.length; i++) {
    job.addResult({"name": names[i]});
  }
}

exports.collect = async (collectible, job) => {
  job.logger().debug('Enter collect', { collectible });
  const quote = "Carpe Diem!"; // Hard coded quote for now.

  // Must return a readable stream from the collect method. Here we are returr
  // the result string wrapped in a Readable.
  const s = new Readable();
```

```
    s.push(JSON.stringify({ host, name: collectible.name, quote }));
    s.push(null);

    // Return readable to stream collected data
    return Promise.resolve(s)
  };
```

4. Next, restart Cribl Stream with the following command:
   `$ /opt/cribl/bin/cribl restart`
5. After Cribl Stream restarts, you must log out and log back in for the new Collector tile to display under Sources.

Configure a New Collector Instance

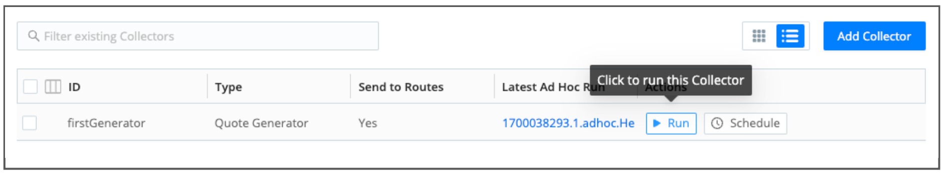
1. In the Cribl Stream UI, navigate to the Quote Generator Collector and click **Add New** to open the **Quote Generator** > **New Collector** modal.
2. Type the following into the **Collector Settings** tab, then click **Save**:
   - **Collector ID**: `firstCollector`
   - **Auto generate names**: `No`
   - **Names**: `Jane, John, Rover`


Sources › Collectors › Quote Generator
New Collector

Collector ID*: firstCollector
Auto generate names: Yes
Names: jane, john, rover
Tags: Enter tags

New Collector Settings

3. On the **Manage Quote Generator Collectors** page click **Run** next to the Collector.

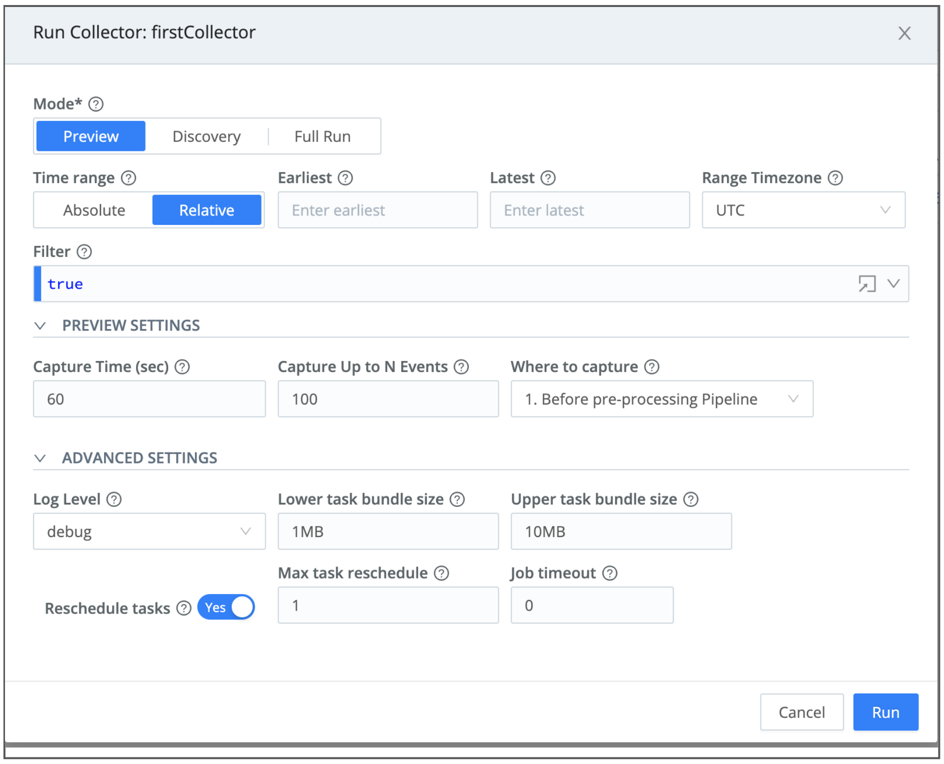


ID	Type	Send to Routes	Latest Ad Hoc Run	Actions
firstGenerator	Quote Generator	Yes	1700038293.1.adhoc.He	Run Schedule

Run the Collector

4. In the **Run configuration** modal:
   - For **Mode**, click **Preview**.
   - For **Log Level**, select **debug**.

- Click **Run** again.



Run Collector: firstCollector

Mode*: [✓ Preview] Discovery | Full Run

Time range: Absolute [✓ Relative]

Earliest: Enter earliest

Latest: Enter latest

Range Timezone: UTC

Filter:

```
true
```

PREVIEW SETTINGS

Capture Time (sec): 60

Capture Up to N Events: 100

Where to capture: 1. Before pre-processing Pipeline

ADVANCED SETTINGS

Log Level: debug

Lower task bundle size: 1MB

Upper task bundle size: 10MB

Max task reschedule: 1

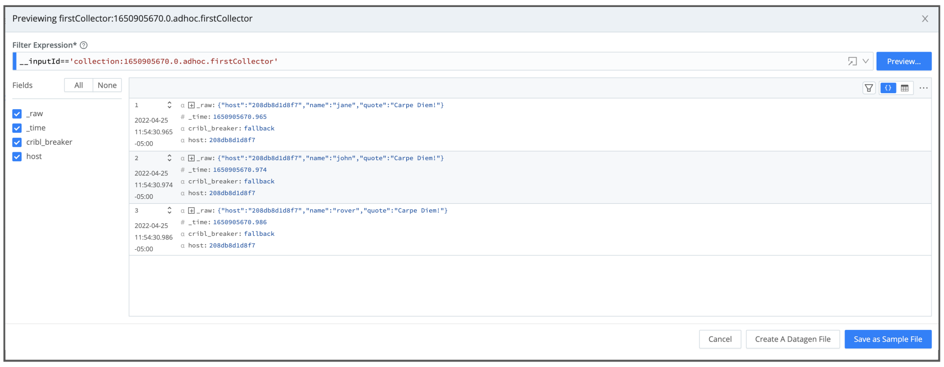
Job timeout: 0

Reschedule tasks: Yes

Cancel | Run

Run Collector Settings

5. If the Collector works, a successful run displays a results dialog, including an event for each name that was added.



Previewing firstCollector:1650905670.0.adhoc.firstCollector

Filter Expression*

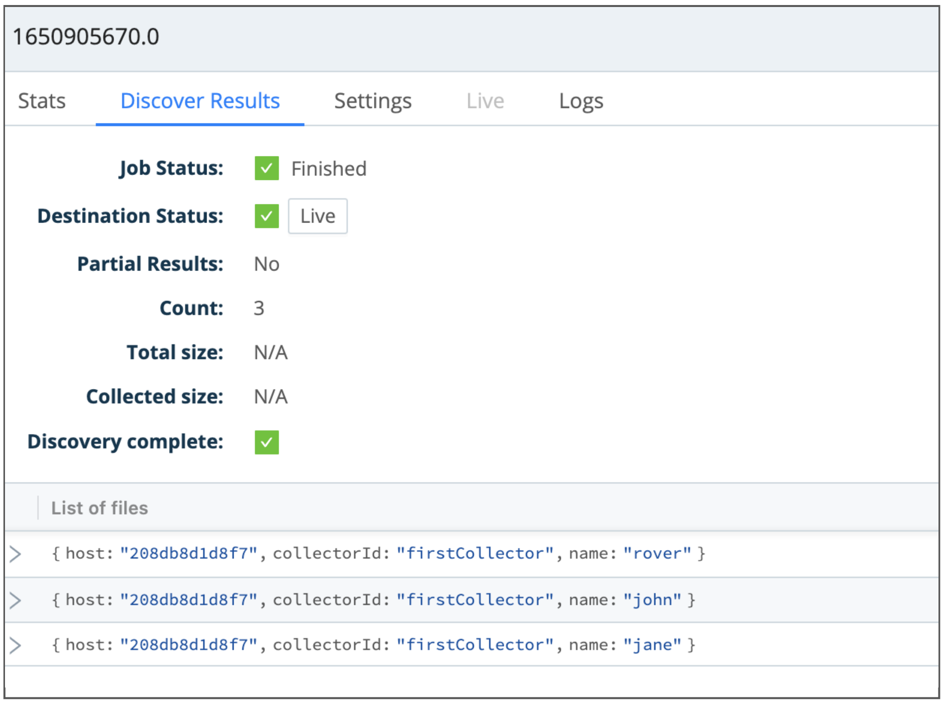
```
__inputId=='collection:1650905670.0.adhoc.firstCollector'
```

Fields: ✓ _raw, ✓ _time, ✓ cribl_breaker, ✓ host

```
1   2022-04-25 11:54:30.965 -05:00
    _raw: {"host":"208db8d1d8f7","name":"jane","quote":"Carpe Diem!"}
    _time: 1650905670.965
    cribl_breaker: fallback
    host: 208db8d1d8f7
2   2022-04-25 11:54:30.974 -05:00
    _raw: {"host":"208db8d1d8f7","name":"john","quote":"Carpe Diem!"}
    _time: 1650905670.974
    cribl_breaker: fallback
    host: 208db8d1d8f7
3   2022-04-25 11:54:30.986 -05:00
    _raw: {"host":"208db8d1d8f7","name":"rover","quote":"Carpe Diem!"}
    _time: 1650905670.986
    cribl_breaker: fallback
    host: 208db8d1d8f7
```

Collector Events

6. Close the **Preview** dialog.
7. On the **Manage Quote Generator Collectors** page, click **Latest ad hoc run**.
8. This will open a dialog from the Job Inspector with information about the run, including job status, items returned by the Discover call, and logs. Click the **Discover Results** tab to see data returned from the Collector’s Discover call:



1650905670.0

Stats | Discover Results | Settings | Live | Logs

Job Status: ✓ Finished
Destination Status: ✓ Live
Partial Results: No
Count: 3
Total size: N/A
Collected size: N/A
Discovery complete: ✓

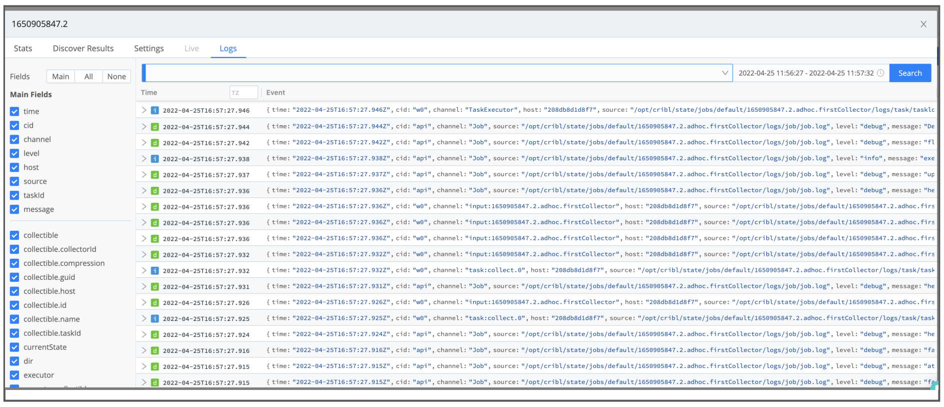
List of files

```
{ host: "208db8d1d8f7", collectorId: "firstCollector", name: "rover" }
{ host: "208db8d1d8f7", collectorId: "firstCollector", name: "john" }
{ host: "208db8d1d8f7", collectorId: "firstCollector", name: "jane" }
```

Discover Results

The Collector method separately. invokes each item returned The Collector argument is the content from each row in the table.

9. Click the **Logs** tab to view logs from the run. This is where you can locate anything logged by `job.logger`.



1650905847.2

Stats | Discover Results | Settings | Live | Logs

Fields: Main | All | None

Main Fields
- ✓ time
- ✓ cid
- ✓ channel
- ✓ level
- ✓ host
- ✓ source
- ✓ taskId
- ✓ message
- ✓ collectible
- ✓ collectible.collectorId
- ✓ collectible.compression
- ✓ collectible.guid
- ✓ collectible.host
- ✓ collectible.id
- ✓ collectible.name
- ✓ collectible.taskId
- ✓ currentState
- ✓ dir
- ✓ executor

2022-04-25 11:56:27 - 2022-04-25 11:57:32 | Search

```
Time                       Event
i 2022-04-25T16:57:27.946  { time: "2022-04-25T16:57:27.946Z", cid: "w0", channel: "TaskExecutor", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/task/tasklc
d 2022-04-25T16:57:27.944  { time: "2022-04-25T16:57:27.944Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "De
d 2022-04-25T16:57:27.942  { time: "2022-04-25T16:57:27.942Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "fl
i 2022-04-25T16:57:27.938  { time: "2022-04-25T16:57:27.938Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "info", message: "exe
d 2022-04-25T16:57:27.937  { time: "2022-04-25T16:57:27.937Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "up
d 2022-04-25T16:57:27.936  { time: "2022-04-25T16:57:27.936Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "he
d 2022-04-25T16:57:27.936  { time: "2022-04-25T16:57:27.936Z", cid: "w0", channel: "input:1650905847.2.adhoc.firstCollector", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firs
d 2022-04-25T16:57:27.936  { time: "2022-04-25T16:57:27.936Z", cid: "w0", channel: "input:1650905847.2.adhoc.firstCollector", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firs
d 2022-04-25T16:57:27.936  { time: "2022-04-25T16:57:27.936Z", cid: "w0", channel: "input:1650905847.2.adhoc.firstCollector", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firs
d 2022-04-25T16:57:27.932  { time: "2022-04-25T16:57:27.932Z", cid: "w0", channel: "input:1650905847.2.adhoc.firstCollector", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firs
i 2022-04-25T16:57:27.932  { time: "2022-04-25T16:57:27.932Z", cid: "w0", channel: "task:collect.0", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/task/task
d 2022-04-25T16:57:27.931  { time: "2022-04-25T16:57:27.931Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "he
d 2022-04-25T16:57:27.926  { time: "2022-04-25T16:57:27.926Z", cid: "w0", channel: "input:1650905847.2.adhoc.firstCollector", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firs
i 2022-04-25T16:57:27.925  { time: "2022-04-25T16:57:27.925Z", cid: "w0", channel: "task:collect.0", host: "208db8d1d8f7", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/task/task
d 2022-04-25T16:57:27.924  { time: "2022-04-25T16:57:27.924Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "he
d 2022-04-25T16:57:27.916  { time: "2022-04-25T16:57:27.916Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "fa
d 2022-04-25T16:57:27.915  { time: "2022-04-25T16:57:27.915Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "at
d 2022-04-25T16:57:27.915  { time: "2022-04-25T16:57:27.915Z", cid: "api", channel: "Job", source: "/opt/cribl/state/jobs/default/1650905847.2.adhoc.firstCollector/logs/job/job.log", level: "debug", message: "fa
```

View Logs

The exceptions are anything logged by our Collector in `init`. Notice that `index.js` defined another logger to use when debugging `init`:

```
const logger = C.util.getLogger('myCollector');
```

You can view the relevant Worker Process' logs on the Monitoring page.

Integrate a Third-Party Package

Next, we'll integrate a third-party package to randomly generate names in the Discover method.

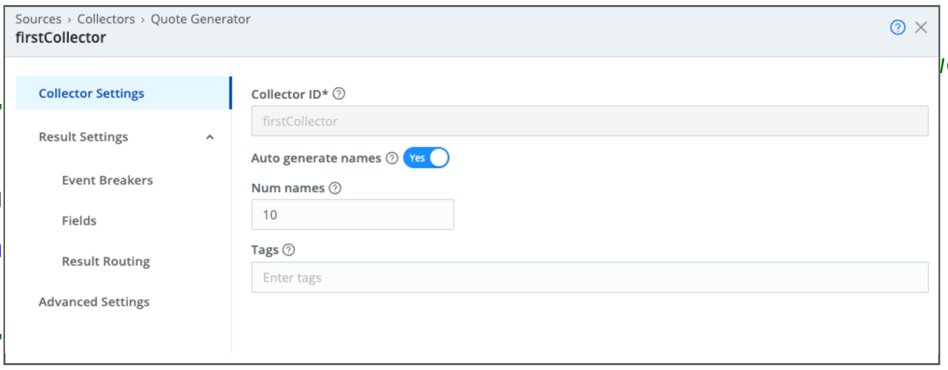
1. In your terminal, follow these steps to install npm (Node Package Manager) into the virtual machine running Cribl Stream:
    - `apt update`
    - `apt install npm`
    - When prompted, select a `Region`.
    - When prompted, select a `Timezone`.
2. Run the following commands in your Collector directory:
    - cd `/opt/cribl/local/cribl/collectors/quote_generator`
    - `npm init`: Answer all questions with default answers. The values are not important for our purposes.
    - `npm install username-generator --save`

   This will install the third-party username-generator package in `/opt/cribl/local/cribl/collectors/quote_generator/node_modules`.
3. Next, update the Collector's `index.js` with the following content, to use this new package to auto-generate names:

```
exports.name = 'Quote Generator';
exports.version = '0.1';
exports.disabled = false;
exports.destroyable = false;

const UsernameGenerator = require('username-generator');
const Readable = require('stream').Readable;
const os = require('os');
const host = os.hostname();
const logger = C.util.getLogger('myCollector'); // For use in debugging init,

let conf;
let batchSize;
let filter;
let autoGenerate = false;
let numNames = 0;
let names;

exports.init = async (opts) => {
  conf = opts.conf;
  //logger.info('INIT conf', { conf });
  batchSize = conf.maxBatchSize || 10;
  filter = conf.filter || 'true';
  autoGenerate = conf.autoGenerateNames
  names = conf.names || [];
  numNames = conf.numNames ?? 3;
  //logger.info('INIT VALUES', { autoGenerate, names, numNames });
  return Promise.resolve();
};

exports.discover = async (job) => {
  if (autoGenerate) {
    // Auto generate usernames using 3rd party package.
    names = [];
    for (let i = 0; i < numNames; i++) {
      names.push(UsernameGenerator.generateUsername('_'));
    }
    job.logger().info('Successfully generated usernames', { numGenerated: name
  }
  for (let i = 0; i < names.length; i++) {
    job.addResult({"name": names[i]});
  }
```

```
{
exports.collect = async (collectible, job) => {
  job.logger().info('Enter collect', { collectible });
  const quote = "Carpe Diem!"; // Hard coded quote for now.

  // Must                                                        we are returr
  // the r
  const s
  s.push(J
  s.push(n

  // Retur
  return Promise.resolve(s)
};
```

4. Now, in Cribl Stream's UI, update the existing Collector to use the auto-generate feature. Add these settings, then click **Save**:
   - **Auto generate names**: Yes
   - **Num names**: 10

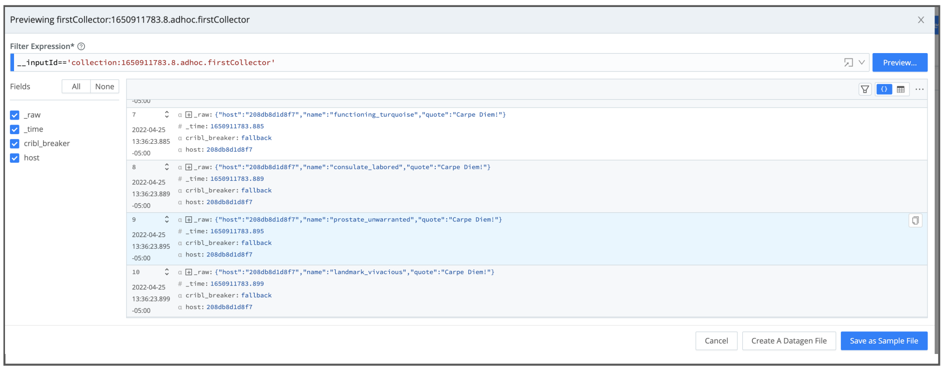

Sources › Collectors › Quote Generator
firstCollector

Collector Settings
Result Settings
Event Breakers
Fields
Result Routing
Advanced Settings

Collector ID*: firstCollector
Auto generate names: Yes
Num names: 10
Tags: Enter tags

Update Existing Collector

5. Run the Collector, notice that 10 Events are now displayed and each username is randomized:


Previewing firstCollector:1650911783.8.adhoc.firstCollector

Filter Expression*
```
__inputId=='collection:1650911783.8.adhoc.firstCollector'
```

Fields: _raw, _time, cribl_breaker, host

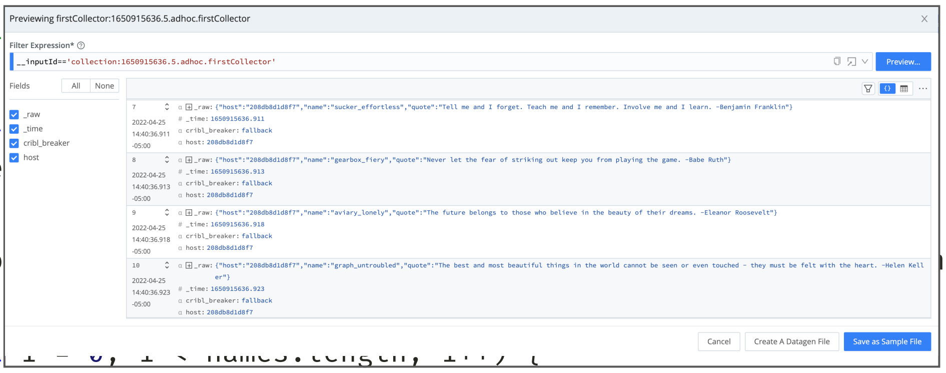
```
-05:00
7           _raw: {"host":"208db8d1d8f7","name":"functioning_turquoise","quote":"Carpe Diem!"}
2022-04-25  _time: 1650911783.885
13:36:23.885 cribl_breaker: fallback
-05:00      host: 208db8d1d8f7
8           _raw: {"host":"208db8d1d8f7","name":"consulate_labored","quote":"Carpe Diem!"}
2022-04-25  _time: 1650911783.889
13:36:23.889 cribl_breaker: fallback
-05:00      host: 208db8d1d8f7
9           _raw: {"host":"208db8d1d8f7","name":"prostate_unwarranted","quote":"Carpe Diem!"}
2022-04-25  _time: 1650911783.895
13:36:23.895 cribl_breaker: fallback
-05:00      host: 208db8d1d8f7
10          _raw: {"host":"208db8d1d8f7","name":"landmark_vivacious","quote":"Carpe Diem!"}
2022-04-25  _time: 1650911783.899
13:36:23.899 cribl_breaker: fallback
-05:00      host: 208db8d1d8f7
```

Update Existing Collector

6. Next, update the Collector's `index.js` with the following content to create a static list of random quotes in the Collect method. You can optionally retrieve a random quote from a REST API instead.

```
exports.name = 'Quote Generator';
exports.version = '0.1';
exports.disabled = false;
exports.destroyable = false;

const UsernameGenerator = require('username-generator');
const Readable = require('stream').Readable;
const os = require('os');
const host = os.hostname();
const { httpSearch, isHttp200, RestVerb } = C.internal.HttpUtils;

const logger = C.util.getLogger('myCollector'); // For use in debugging init,

// Quotes to randomize
const quotes = [
  "Spread love everywhere you go. Let no one ever come to you without leaving
  "When you reach the end of your rope, tie a knot in it and hang on. -Frankli
  "Always remember that you are absolutely unique. Just like everyone else. -M
  "Don't judge each day by the harvest you reap but by the seeds that you plar
  "The future belongs to those who believe in the beauty of their dreams. -Ele
  "Tell me and I forget. Teach me and I remember. Involve me and I learn. -Ber
  "The best and most beautiful things in the world cannot be seen or even touc
  "It is during our darkest moments that we must focus to see the light. -Aris
  "Whoever is happy will make others happy too. -Anne Frank",
  "Do not go where the path may lead, go instead where there is no path and le
  "You will face many defeats in life, but never let yourself be defeated. -Ma
  "The greatest glory in living lies not in never falling, but in rising every
  "In the end, it's not the years in your life that count. It's the life in yc
  "Never let the fear of striking out keep you from playing the game. -Babe Ru
  "Life is either a daring adventure or nothing at all. -Helen Keller",
  "Many of life's failures are people who did not realize how close they were
  "You have brains in your head. You have feet in your shoes. You can steer yc
  "If life were predictable it would cease to be life and be without flavor. -
  "In the end, it's not the years in your life that count. It's the life in yc
  "Life is a succession of lessons which must be lived to be understood. -Ralp
  "You will face many defeats in life, but never let yourself be defeated. -Ma
]

let conf;
let batchSize;
let filter;
let autoGenerate = false;
let numNames = 0;
let names;

exports.init = async (opts) => {
  conf = opts.conf;
  //logger.info('INIT conf', { conf });
  batchSize = conf.maxBatchSize || 10;
  filter = conf.filter || 'true';
  autoGenerate = conf.autoGenerateNames
  names = conf.names || [];
  numNames = conf.numNames ?? 3;
  //logger.info('INIT VALUES', { autoGenerate, names, numNames });
  return Promise.resolve();
};

exports.discover = async (job) => {
  if (autoGenerate) {
    // Aut
    names
    for (l
      name
    }
    job.lo                                                     erated: name
  }
  for (let i = 0; i < names.length; i++) {
    await job.addResult({"name": names[i]});
  }
}

exports.collect = async (          le, job) => {
  const quote = quotes[Math.trunc(Math.random()*100)%quotes.length];

  const s = new Readable();
  s.push(JSON.stringify(result));

  // Return readable to stream collected data
  return Promise.resolve(s)
};
```

7. Run the Collector to randomly retrieve the quotes from the list:



```
Previewing firstCollector:1650915636.5.adhoc.firstCollector
Filter Expression*
__inputId=='collection:1650915636.5.adhoc.firstCollector'

7   2022-04-25 14:40:36.911 -05:00
    _raw: {"host":"208db8d1d8f7","name":"sucker_effortless","quote":"Tell me and I forget. Teach me and I remember. Involve me and I learn. -Benjamin Franklin"}
    _time: 1650915636.911
    cribl_breaker: fallback
    host: 208db8d1d8f7
8   2022-04-25 14:40:36.913 -05:00
    _raw: {"host":"208db8d1d8f7","name":"gearbox_fiery","quote":"Never let the fear of striking out keep you from playing the game. -Babe Ruth"}
    _time: 1650915636.913
    cribl_breaker: fallback
    host: 208db8d1d8f7
9   2022-04-25 14:40:36.918 -05:00
    _raw: {"host":"208db8d1d8f7","name":"aviary_lonely","quote":"The future belongs to those who believe in the beauty of their dreams. -Eleanor Roosevelt"}
    _time: 1650915636.918
    cribl_breaker: fallback
    host: 208db8d1d8f7
10  2022-04-25 14:40:36.923 -05:00
    _raw: {"host":"208db8d1d8f7","name":"graph_untroubled","quote":"The best and most beautiful things in the world cannot be seen or even touched - they must be felt with the heart. -Helen Keller"}
    _time: 1650915636.923
    cribl_breaker: fallback
    host: 208db8d1d8f7
```

Run the Collector

8. In the next few steps, you'll back up the files, by opening a shell window on your local machine and running the indicated commands.
9. List Docker containers by running `docker ps`, as shown here:

```
$ docker ps
CONTAINER ID   IMAGE                  COMMAND                  CREATED
208db8d1d8f7   cribl/cribl:latest    "/sbin/entrypoint.sh…"   6 hours ago   Up
```

In this example, the **Container ID** is 208db8d1d8f7. You'd paste this into the next commands.

10. Run: `docker exec -it 208db8d1d8f7 tar cvzf /tmp/container_source.tgz /opt/cribl/local/cribl/collectors/quote_generator`
11. Run: `docker cp 208db8d1d8f7:/tmp/container_source.tgz`

You can locate the archive file containing the source code in the current directory. After you back up the source code, it is safe to shut down the Docker container.

Building a Collector (Distributed Environment)

When you create a Collector in a distributed environment, the directory path is slightly different from the single–instance example above. E.g., for the `default group`, you should create Collectors on the Leader Node in the directory: `opt/cribl/groups/default/local/cribl/collectors`.

The format of the path is: `/opt/cribl/groups/<workerGroup>/local/cribl/collectors`

For example, assuming you want to create a new Collector in group is `myGroup`, the path would be: `/opt/cribl/groups/myGroup/local/cribl/collectors`.

You'd proceed as follows:

1. Connect to the Leader Node.
2. `cd $CRIBL_INSTALL/groups/default/local/cribl`
   `$CRIBL_INSTALL` refers to the directory where Stream is running, e.g.: `/opt/cribl`
3. `mkdir collectors`
4. `cd collectors`
5. `mkdir quote_generator`
6. `cd quote_generator`
7. `cp ../../../../default/cribl/collectors/script/* .`

Building the Collector in a distributed environment works the same as in single-instance environment with exception of a few differences:

1. You must add the files to a Worker–Group directory, as described above.
2. You must commit and deploy changes to the Workers.

Given the extra steps, we recommend that you first build the Collector in a standalone environment, before deploying it to a distributed environment.

17.6. Integrating With Other Services

17.6.1. Amazon

17.6.1.1. Amazon S3 Better Practices

The “better practices” in this guide all apply to Cribl Stream’s Amazon S3-based Sources and Destinations. Many also apply to other object stores, like Azure Blob Storage, MinIO, and Google Cloud Storage.

View the AWS S3 with Cribl Best Practices video presentation from Cribl Community Office Hours.

Foundations: Cribl Stream and Object Storage

In this first section, we’ll lay out four basic factual underpinnings:

- How Cribl writes to object storage.
- Writing to a Cribl Amazon S3 Destination.
- Reading from Cribl’s Amazon SQS Source.
- Replaying from Amazon S3 buckets, via Cribl’s S3 Collector.

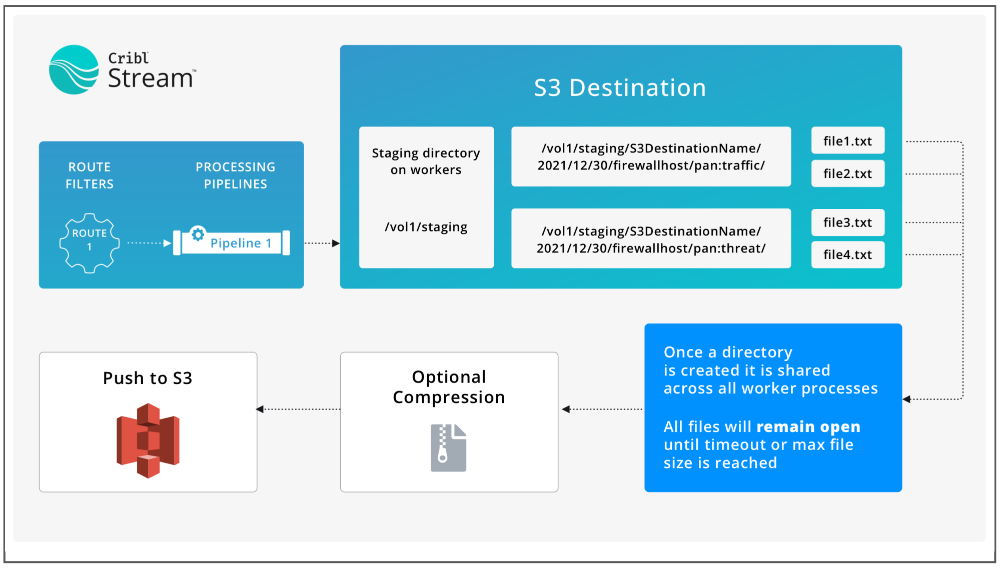
Then, this guide will go on to detail 10 “better practices” (and a bonus one!) as we cover four broad topics:

- Writing to object storage, generically.
- Cardinality and partitioning expressions.
- Amazon S3–specific optimization.
- Better partitioning expressions.

How Cribl Writes to Object Storage

Cribl does not write data **directly** to its object storage Destinations, which include Amazon S3; any solution with an S3 API; and other object storage Destinations, like Azure Blob Storage, MinIO, and Google Cloud Storage.

Data persists locally to a “staging location” on the Workers. The Worker Processes append new data to the staging location based on partitioning expressions and configurations such as file size and max timeout settings. Each Worker Process, governed by its own specific settings, creates and maintains its own set of files.


Cribl Stream
S3 Destination
ROUTE FILTERS
PROCESSING PIPELINES
ROUTE 1
Pipeline 1
Staging directory on workers
/vol1/staging
/vol1/staging/S3DestinationName/2021/12/30/firewallhost/pan:traffic/
/vol1/staging/S3DestinationName/2021/12/30/firewallhost/pan:threat/
file1.txt
file2.txt
file3.txt
file4.txt
Once a directory is created it is shared across all worker processes
All files will remain open until timeout or max file size is reached
Optional Compression
Push to S3

Writing to object storage – example

Writing to a Cribl Amazon S3 Destination

When writing to our Amazon S3 Destination, or any object storage, consider two main settings: **File name prefix expression** and **Partitioning expression**.

Cribl allows a maximum of 2000 open files per Worker Process per S3 Destination. On a system with 14 Worker Processes, that translates to a maximum of 28,000 open files per S3 Destination. On a system with 30 CPUs, that translates to 60,000 open files.

When a specific Worker Process hits the maximum open files, it closes all open files, moves them to the final output location, and creates new ones. If you have multiple object storage Destinations configured in Cribl, ensure that the total maximum open files across all these Destinations is less than your system's configured maximum open files. To figure out this maximum, multiply the number of file-based Destinations by max open file limit for each such Destination and by number of Worker Processes (Destinations x max open file limit x Worker Processes).

The default maximum open file settings on many Linux systems are too low for Cribl Stream when writing to an object storage Destination. You'll need to increase your system max open files and/or process max open files settings, as explained below.

Additionally, to limit the number of files in each directory, **Partitioning expression** should specify dates down to the hour or minute.

Reading from Cribl's Amazon SQS Source

When reading data using Cribl Stream's Amazon SQS Source, a few settings can greatly speed up data retrieval. These are:

- **General Settings** > **Queue**, a.k.a. the filename filter.
- **Advanced Settings** > **Max messages**.
- **Advanced Settings** > **Num receivers**.

Even more importantly, data retrieval speed is also governed by Destination settings and Pipeline efficiency.

Replaying from Amazon S3 Buckets

When replaying from S3 via Cribl's S3 Collector, optimize your performance by adjusting the following:

- **Optional Settings** > **Path**, a.k.a. the filter.
- Pipeline efficiency.
- Destination settings.
- Destination health.

The **Path** should specify the time down to the hour, or in some cases, down to the minute.

Also, consider the API limits for your object storage bucket. AWS sets a limit of 3,500 writes/second and 5,500 reads/second per S3 prefix. An S3 prefix is a path on Amazon S3, including bucket name and any subdirectories up to but not including the object name. The S3 prefix always ends in (and includes) the slash before the object name. In its simplest form, the S3 prefix is just `BucketName/`.

AWS API limits apply to all attempted access to these prefixes, cumulatively. You probably won't hit the write limits while putting the "better practices" in this guide into effect. However, populating too many files in any one directory might trigger throttling when reading and performing replays.

Some Cribl Destinations have **File prefix expression** and/or **Partitioning expression** settings. Do not confuse these with the AWS prefix.

Writing to Object Storage, Generically

When you deploy a Cribl Amazon S3 Compatible Stores Destination, start with a simple global **Partitioning expression** and the default **File name prefix expression**.

This will work as long as cardinality remains below 2,000 during the max timeout period. Also, apply the procedures to increase the system max open file settings, as described later in this guide.

Better Practice 1: Mind the Partitioning Expression

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')
/${index ? index : 'no_index'}
/${host ? host : 'no_host'}
/${sourcetype ? sourcetype : 'no_sourcetype'}
```

If cardinality exceeds 2,000, you can either:

- Remove the less-important fields from the expression; or
- Create different partitioning expressions for different event types, categories, etc.

On the **Advanced Settings** tab:

- Match the **Max open files** to cardinality, without exceeding 2,000.
- Toggle **Remove staging dirs** to `Yes`.
- Set the **Storage class** to `Intelligent Tiering`.

Still in **Advanced Settings**, keep the default settings for:

- **Max file size (MB)**: `32`.
- **Max file open time (sec)**: `300`.
- **Max file idle time (sec)**: `30`.

As you gain deeper understanding of the data, consider the following options:

- Create a set of different S3 Destinations, each with its own **Partitioning expression**. They can all point to the same S3 bucket. Be aware that doing so will increase the number of API requests becuase events will be spread over more files.
- If the numbers of open files becomes too large, discuss alternative architectures (e.g., creating additional Worker Groups) with your Cribl Account team or Support.

Better Practice 2: Optimize System Settings for Max Open Files

Consider the workflow for writing to Cribl S3 Destinations:

- Each Cribl Worker Process creates its own set of files in a staging directory based on **Partitioning expression** and **File name expression**.
- The maximum partitioning expression cardinality allowed is 2,000. This translates to a maximum of 2,000 open files per Worker Process per object storage Destination.

- Each Destination writes files to its staging directory until the number of files exceeds the configured maximum, or the Destination reaches any of the following limits: **Max file size (MB)**, **Max file idle time (sec)**, or **Max file open time (sec)**.

When writing to an object store, more Worker Nodes with fewer CPUs is better than fewer Worker Nodes with more CPUs (e.g., 5 Workers with 12 vCPUs is better than 2 systems with 32 vCPUs).

You should apply this principle to realize two benefits:

- To maximize the number of open files.
- To have more S3 Destinations with more precisely-focused partitioning expressions.

Linux defaults for maximum open files tend to be too low for writing to Cribl S3 Destinations, but the 65,536 limit recommended by Cribl is a good starting point. When setting the maximum open files substantially higher than 65,536, you should track system health.

You can further increase the maximum number of open files allowed by your system based on the number of S3 Destinations you are planning to use and the number of Worker Processes on each Node. The table below provides some examples:

# of S3 Destinations in Cribl	Max cardinality per Dest.	Max open files in Cribl Dest.	# Worker Process on each Worker Node	Min Linux per-process open file limit	Min Linux system plus Cribl user open file and 1k reserved for the system
1	2,000	2,000	10	3,000	21,000
1	2,000	2,000	14	3,000	29,000
1	2,000	2,000	22	3,000	45,000
2	2,000	2,000	10	5,000	41,000
2	2,000	2,000	14	5,000	57,000
2	2,000	2,000	22	5,000	89,000
5	2,000	2,000	10	11,000	101,000
5	2,000	2,000	14	11,000	141,000
5	2,000	2,000	22	11,000	221,000

Better Practice 3: Update Max Open File Settings on Hosts

Here are basic instructions that cover many Linux distributions. (Check the documentation for your specific Linux distribution and version for any variations required.)

Edit the following settings. You will be required to reboot.

```
# Update `fs.file-max` to a very large value run (as root):
sysctl -w fs.file-max=<open-files-limit>

# or edit sysctl.conf (as root):

vi /etc/sysctl.conf
fs.file-max = <open-files-limit>


# vi /etc/security/limits.conf (as Cribl user)
* soft nproc 65536
* hard nproc 65536
* soft nofile 65536
* hard nofile 65536
```

After you reboot, verify the settings:

```
# sysctl -p
# cat /proc/sys/fs/file-max
# sysctl fs.file-max
```

Cardinality and Partitioning Expressions

As we've seen, partitioning expressions and file expressions interact to determine cardinality. In this section, we'll examine this interaction more closely and learn better practices based on what we find.

Where Cardinality Begins

Partitioning expressions specify the directory structure that Cribl adds when writing to an object storage Destination.

For example, the following partitioning expression extracts the `year`, `month`, `day`, `hour`, `minute`, `index`, `host`, `sourcetype`, `server_ip`, `status_code`, and `Method`, and populates them as the directory structure.

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')
/${index ? index : 'no_index'}
/${host ? host : 'no_host'}
/${sourcetype ? sourcetype : 'no_sourcetype'}
/${server_ip}
/${status_code}
/${Method}/`
```

A file expression prefix comes into play next. The default file expression is just a random file name. However, an expression like this would create files that also include the hour and minute in each file name. This reflects the time the file was created, not the time of each event.

```
${C.Time.strftime(_time ? _time : Date.now() / 1000, '%H%M')}
```

Together, the partitioning expression and the file expression must not produce a cardinality greater than 2,000.

Calculating Cardinality

Cardinality is the quantity of combinations of unique values, within a time period, for each parameter in the expressions.

For example, a partitioning expression could specify the following:

- Within an hour.
- 5 unique sourcetype values.
- 10 unique host values.
- 2 unique source values for each host.

For this example, we could calculate the maximum cardinality as `(5*10*2)=100`.

It's important to also consider the timeframe this is calculated within. That cardinality might be 100 over a 1-minute period, but might be 500 over a 5-minute period.

If we further add a file name prefix expression, this increases the cardinality. For example, let's add the method and top-level URI as file name prefix parameters. If there are 3 possible methods, and 10 URIs, the cardinality just went up from 100 to `100*3*10=3,000`.

Cardinality has the greatest impact on how you should configure an object storage Destination in Cribl Stream. For every combination of partitioning expressions, file prefix expressions, and Worker Processes, there can be an open file on the Worker Node's system.

A partitioning expression and file prefix expression should produce fewer than 2,000 potential combinations. When a Worker Process exceeds the configured **Max open files**, this generates errors; all open files for that Worker Process are simultaneously closed; and, new files will be created.

The Worker Process might experience some extra load while all files are processed. If this occurs infrequently, that's generally okay. But if it happens several times per hour, you should consider how to improve the combination of partitioning and file expressions.

As a starting point, consider these two strategies:

1. Write to one S3 bucket, incorporating parameters common to each event.

For example, here's a partitioning expression that specifies year, month, day, hour, minute, index, sourcetype, source, and host:

```
${C.Time.strftime(_time ? _time:Date.now()/1000,'%Y/%m/%d/%H/%M')/${index}/${host},
```

2. Dedicate different S3 Destinations to different event types. During replays, this makes searching specific data faster. However, this benefit comes at the expense of needing a larger number of open files. Should you pursue this option, test large numbers of open files on a test system in your environment. Such testing is especially crucial in virtualized environments with shared disk, CPU, and memory resources.

Better Practice 4: Select Files for Partitioning and File Expressions

The main factors to consider when selecting fields for a partitioning or filename prefix expression are:

- Configure Time-based filtering: All partitioning expressions should include `year`, `month`, `day`, `hour`, and optionally, `minute`. This facilitates filtering during replays. It also limits the files in any one directory, so that replays are not throttled when returning large volumes of files.
- Identify Universal fields: Fields that users will typically search on during replays across all datasets. Examples include `sourcetype`, `host`, `source`, `category`, and `index`.
- Optionally, configure specific fields for different datasets.
  Examples include:

    - Web logs: server IP, method, status code.
    - Firewall logs: source zone, destination zone, accept/deny status.
    - Flow logs: protocol, accept/deny status.

  Be careful when pursuing this option as the number of open files can grow exponentially with too many multiple partition and file name prefix expressions.

Analyzing Partitioning Expressions

Let’s work through some examples of partitioning expressions, with hypothetical possible values for each parameter, to see what cardinalities they produce.

Example 1

This one produces reasonable results.

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')}
/${sourcetype}/${host}/${AttackType}`
```

Possible values:

- `sourcetype`: 5.
- `host`: 100.
- `AttackType`: 5.

Potential cardinality: `5 x 100 x 5 = 2,500`.

Example 2

Here, too, cardinality is kept commendably low.

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')}/${state}/${ir
```

Possible values:

- `state`: 2.
- `interface`: 50.
- `dest_ip`: 20.

Potential cardinality: `2 x 100 x 10 = 2,000`.

Example 3

This is a bad partitioning expression which produces out-of-control cardinality. Back to the drawing board!

```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')}/${state}/${in
```

Possible values:

- `state`: 2.
- `interface`: 100.
- `src_ip`: 65,535.

Potential cardinality: `2 x 100 x 65,535 = 13,107,000`.

For more examples like these, see Better Partitioning Expressions.

Seeing How Cardinality Changes Over Time

In the examples so far, we've assumed typical expected values for parameters. Another useful exercise is to estimate the maximum possible values of each parameter, and multiply each value to obtain maximum theoretical cardinality.

You should also investigate how cardinality changes depending on when and how long you measure it. To do this, query/search your analytic tool to determine cardinality for different durations over several days.

Run the searches over different duration spans (e.g., 1 minute, 3 minutes, 5 minutes) to determine the best timeout period. For example, your cardinality might be 2,000 over a 2-minute span, but 10,000 over a 5-minute span. In this case, it is most certainly beneficial to set the maximum file timeout to 2 minutes.

Below are examples of Splunk searches over different span periods, with maximum open file time set to 3 minutes. Change the fields to the ones you'd prefer to incorporate into a partition and/or file name prefix expression, and try running them in your environment.

Here's an example using a "traditional" search method:

```
earliest=-24h index=*
|bin _time span=3m
| stats dc(source) as dc by host, sourcetype _time
| stats sum(dc) by _time
```

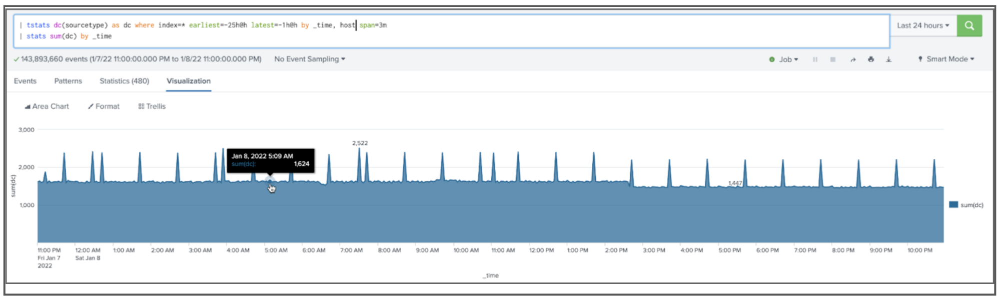
Here's an example using `tstats`:

```
| tstats dc(source) as dc where index=* earliest=-25h@h latest=-1h@h by _time, hos
| stats sum(dc) by _time
```

The `tstats` search produces the results shown in the diagram below.

- Cardinality mostly stays below 1,625.
- There are some hourly peaks around 2,500. These spikes, which exceed the max allowed cardinality of 2,000, are infrequent.

All of this is generally acceptable.



```
| tstats dc(sourcetype) as dc where index=* earliest=-25h@h latest=-1h@h by _time, host span=3m
| stats sum(dc) by _time
```

✓ 143,893,660 events (1/7/22 11:00:00.000 PM to 1/8/22 11:00:00.000 PM)

cardinality Check Results

Better Practice 5: Size a Separate Fast(er) Disk for Staging Location

The system's ability to handle enough open files is a key factor that drives performance. This requires:

- Setting **Max open files** high enough.
- Ensuring that the staging directory on the Workers can handle simultaneous writing tens of thousands of open files. The metric to use here is Input/Output Operations per Second (IOPS). A good starting point is 2,000 IOPS.

Cribl recommends that you allocate a separate disk for the staging directory on Workers, especially when you have configured multiple object storage Destinations. Follow these general, conservative guidelines:

- Toggle **Advanced Settings** > **Remove staging dirs** to `Yes`. This means that any empty directory that has not been written to for an hour, is deleted.
- Disk size of 500 GB should be adequate, since the Workers need only enough disk capacity for a few minutes' worth of data. 500 GB errs on the side of caution, allowing us to store as much as 4 hours of data to disk. See the formula and examples below.
- Because Cribl writes to the staging directory only when the object storage Destinations are healthy, disk size does not usually become an issue. If the Worker encounters or triggers a backpressure

situation, it stops writing to the staging directory.

- The staging directories are not intended for high availability, but rather to support creation of large-enough files to write to object storage.

Replay from a dedicated Worker Group. Do not compete for resources against production, real-time, streaming resources to perform replays!

Determining Staging Directory Disk Size

You can calculate an optimal size for your staging directory disk using the following formula:

- `Data Volume` (GB/day) you expect to ingest, times `1.25` metadata factor; divided by
- `1440` (minutes in a day), times
- `Duration` (minutes) you want to store data on disk; divided by
- `Number of Workers` writing to a staging directory.

The result will be disk size per Worker.

The figure below illustrates the formula.

$$\frac{Data\ Volume\ (GB/day) \times 1.25\ metadata\ factor}{1440\ mins/day} \times\ mins\ to\ store\ data\ to\ disk\ \div\ number\ of\ Workers$$

Here are some examples:

Data Volume	Number of Workers	Duration	Disk Size per Worker
1 TB/day	3	2	35 GB
10 TB/day	5	4	417 GB
50 TB/day	45	4	232 GB

Amazon S3 Optimization

Once you configure your Amazon S3 permissions, verify access, and (if applicable) discover your Cribl.Cloud Organization's AWS ARN, you can start applying some more better practices.

Amazon S3 Permissions

If your Workers are hosted outside AWS, you will need an access key and secret key to access the S3 bucket. If the Workers are EC2 instances or in AWS EKS, it's most ideal to assign an EC2 service role to the instances. The role will need access to the S3 bucket. For policies for reading and writing to/from Amazon S3, see AWS Cross-Account Data Collection.

Here's a policy to grant a role, or user access, with read and write permissions the S3 bucket. Just change the bucket name in this policy:

```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Action":
     [
        "s3:GetObject",
        "s3:PutObject"
      ],
     "Resource": "arn:aws:s3:::bucketname/*"
   },
   {
     "Effect": "Allow",
     "Action":
       [
            "s3:ListBucket",
               "s3:GetBucketLocation"
       ],
     "Resource": "arn:aws:s3:::bucketname"
   }
 ]
}
```

If `assumeRole` is required, here's a sample policy to assume the role that has access to the S3 bucket:

```
{
"Version": "2012-10-17",
"Statement": [
  {
    "Effect": "Allow",
    "Principal": {
      "AWS": "arn:aws:iam::111111111111:role/account-a-cribl-stream-assumerole-rol
    },
    "Action": "sts:AssumeRole",
    "Condition": {
      "StringEquals": {
        "sts:ExternalId": "cribl-s3cre3t"
      }
    }
  }
]
}
```

Test your policy with AWS's IAM policy simulator to ensure there are no boundary policies that might conflict with this policy.

Verifying S3 Access

To troubleshoot access to your S3 bucket, install the AWS CLI on one of the Workers. Log in using your AWS user and Region. Assume the role if necessary. Then, attempt to download a file from the bucket or write a new file to the bucket.

Example of assuming a role:

```
aws sts assume-role --role-arn "arn:aws:iam::123456789012:role/example-role" --rol
```

Example of validating the role you'll use to run subsequent commands:

```
aws sts get-caller-identity
```

Example of listing files in a bucket:

```
aws s3 ls s3://s3-bucket-name
```

Example of sending stdin to a specified bucket:

```
aws s3 cp - <target> [--options]
```

Example of writing a local file to the bucket:

```
aws s3 cp localfilename.txt s3://bucket-name
```

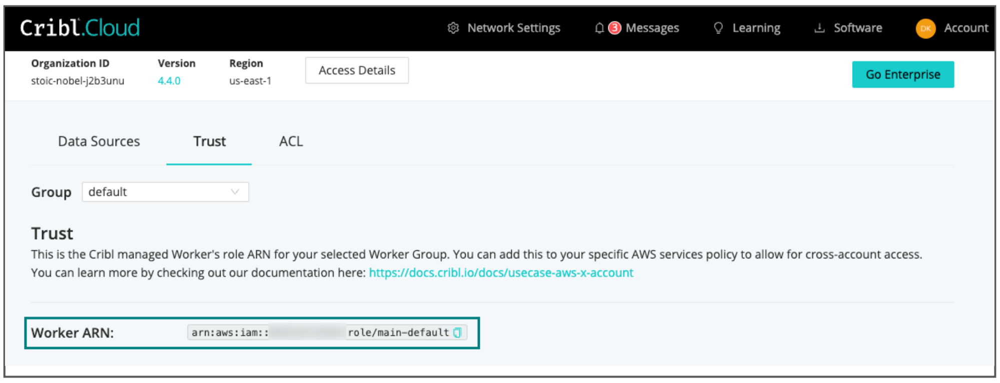
Example of reading a file from S3:

```
aws s3 cp s3://bucket-name localfilename.txt
```

Accessing Amazon S3 from Cribl.Cloud

Every Cribl.Cloud Organization exposes its own unique AWS ARN.

From the main Cribl.Cloud portal, click the **Network Settings** link, then select the **Trust** tab to expose the Cribl.Cloud ARN.



Organization ID stoic-nobel-j2b3unu
Version 4.4.0
Region us-east-1

Group default

Trust
This is the Cribl managed Worker's role ARN for your selected Worker Group. You can add this to your specific AWS services policy to allow for cross-account access.
You can learn more by checking out our documentation here: https://docs.cribl.io/docs/usecase-aws-x-account

Worker ARN: arn:aws:iam::[redacted]role/main-default

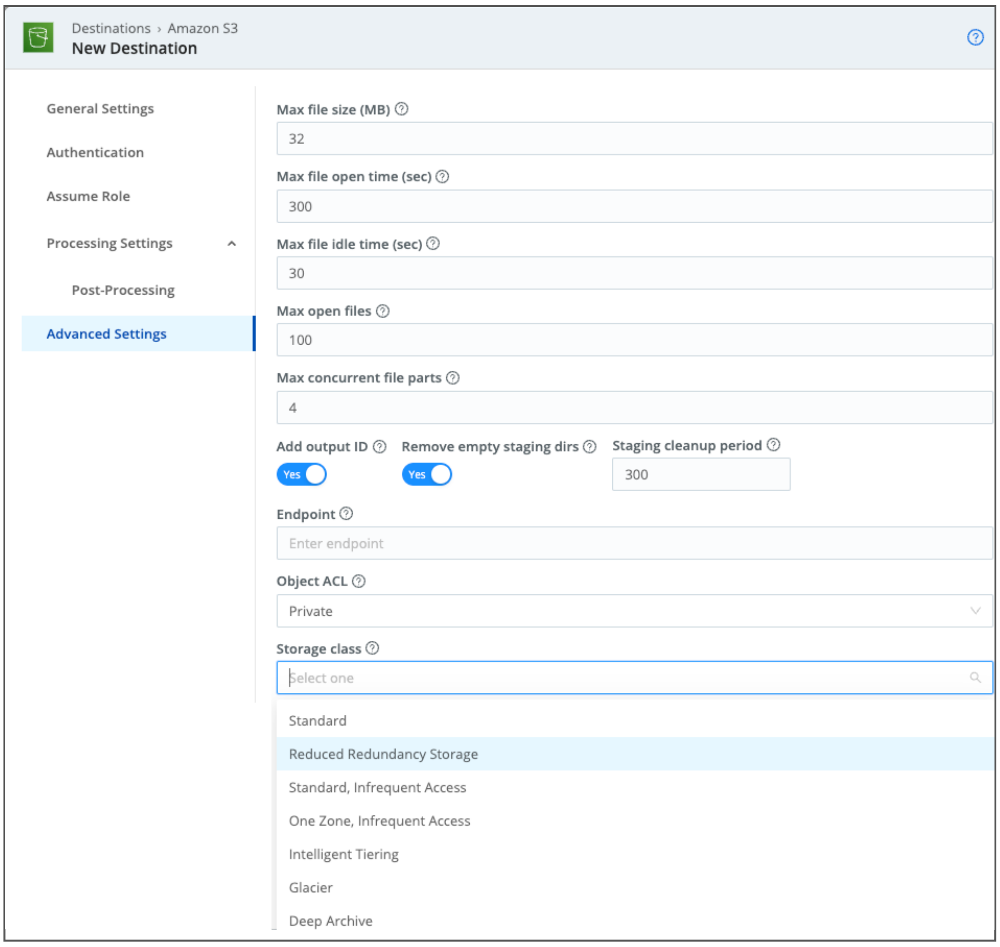
Cribl.Cloud ARN

Configure your AWS account with access to the S3 bucket, to allow Cribl.Cloud to assume that role, as illustrated above in Amazon S3 Permissions.

Better Practice 6: Tuning Amazon S3 Destination Settings

Under the Amazon S3 Destination's **General Settings** and **Advanced Settings** tabs are several settings for tuning your Amazon S3 Destination. Below are some recommendations for tuning those settings, in order of importance:

1. **Staging Location**: Change this to a directory on the Workers with increased IOPS, to accommodate thousands or tens of thousands of simultaneous open files. (Cannot be configured on Cribl.Cloud-managed Workers.)
2. **Remove staging dirs**: This setting must be enabled, so as not to leave millions of empty directories on the system over time.
3. **Compress**: Always enable compression, to reduce the size of files uploaded and downloaded from S3. This can reduce your S3 bill by 8–15x. Note that compression does add to the processing time – but transfer time will be substantially faster.
4. **Max open files**: Adjust this to be slightly larger than the data's cardinality, but not greater than 2,000. Also, ensure that the system can support the number of open files for all Worker Processes, and across all configured S3 Destinations. This value is set per process. 32CPUs*100 - 3200 max open files. Increase this value to avoid "Too many open files" errors. Default open file limit on Linux is 1024/process.
5. **Max file size (MB)**: If you have a separate S3 Destination for high-volume data sources, such as VPC Flow Logs or firewall logs, consider raising this value in order to create fewer files. Start with 64 MB. Otherwise, the default 32 MB setting is generally acceptable.
6. **Max file open time (sec)**: First, determine your cardinality over 1 minute, 3 minutes, 5 minutes, and 10 minutes. Adjust this setting to match the lowest-cardinality period. E.g., if cardinality is 10,000 over 5 minutes, but 2,000 over 90 seconds, change this setting to 90 seconds. If cardinality is 2,000 over 5 minutes, then the default 5 minutes (300 seconds) is acceptable.
7. **Max idle time (sec)**: The default 30 seconds is generally acceptable. In cases with high cardinality, lowering this setting can result in fewer open files when processing bursty data.
8. **Data format**: If using a Passthru Pipeline, change this setting to `Raw`. Otherwise, leave it at the JSON default to preserve all metadata added to the events.
9. **Storage class**: For Amazon S3 Destinations, consider the "infrequent access tier" or "intelligent tier" to reduce overall cost if the data will be rarely searched. Examples of rarely-searched data include threat-hunting or compliance-reporting use cases. "Standard, Infrequent Access" and "Intelligent Tiering" options should be used on low-access use cases such as threat hunting or compliance audit.
10. **AWS Region**: Leave this field blank for non-AWS Destinations, unless instructed otherwise. For AWS, set this to the nearest geographic location to the Workers.


Destinations › Amazon S3
New Destination

Max file size (MB): 32
Max file open time (sec): 300
Max file idle time (sec): 30
Max open files: 100
Max concurrent file parts: 4
Add output ID: Yes
Remove empty staging dirs: Yes
Staging cleanup period: 300
Endpoint: Enter endpoint
Object ACL: Private
Storage class: Select one

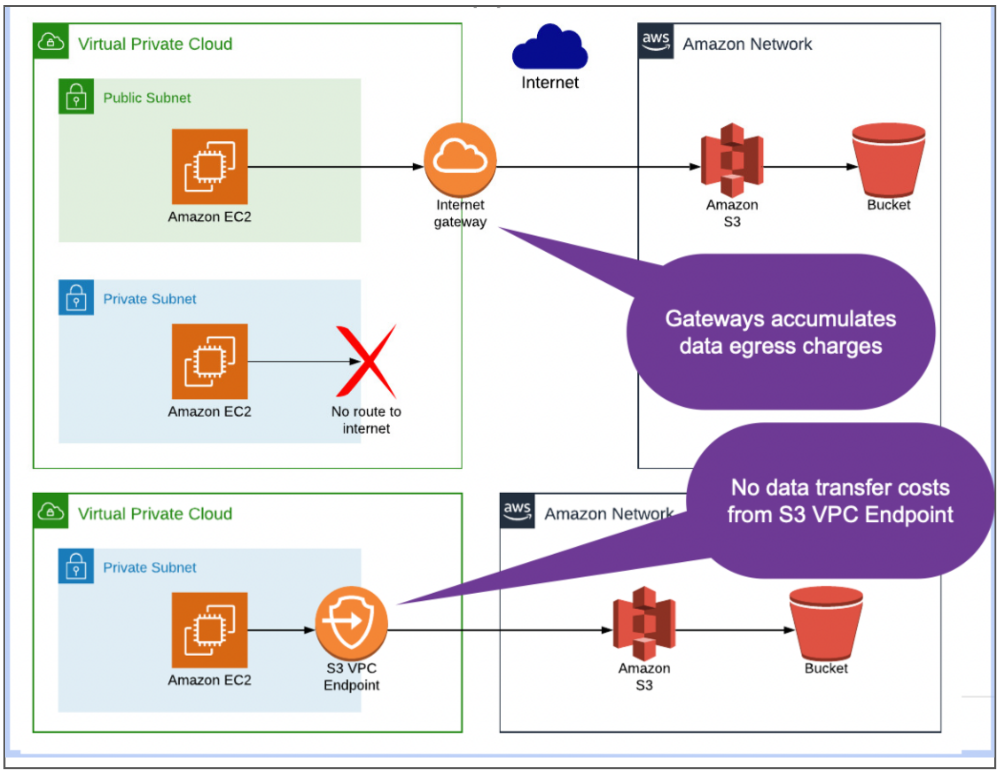
Standard
Reduced Redundancy Storage
Standard, Infrequent Access
One Zone, Infrequent Access
Intelligent Tiering
Glacier
Deep Archive

Configuring S3 Destination Settings

Better Practice 7: Lower AWS Costs

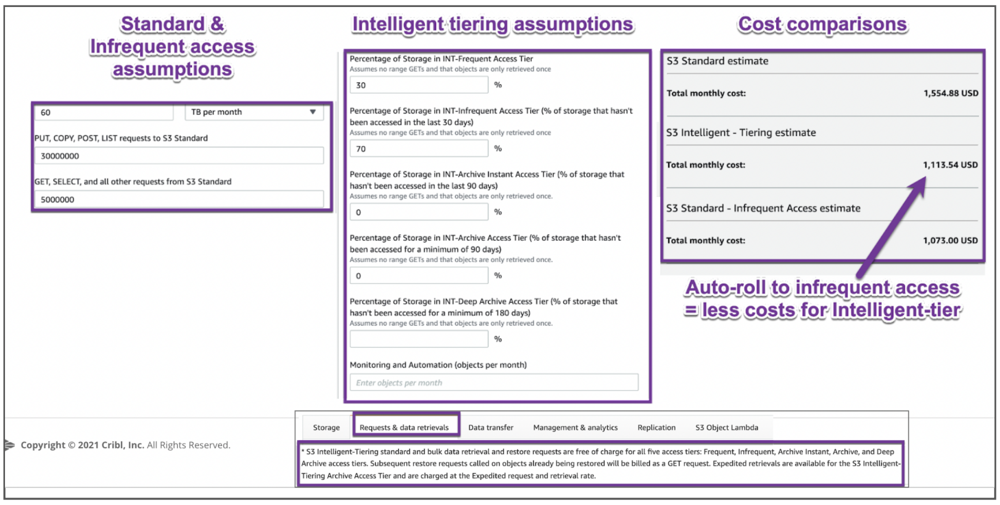
Lowering AWS costs can be achieved by a few means. First and foremost, ensure that compression is enabled on your Cribl S3 Destination and set to `gzip`. This will reduce both egress costs and S3 storage costs.

To eliminate egress costs from your Workers to S3, map a VPC endpoint on the S3 bucket to the VPC where your Workers reside.



Mapping a VPC endpoint to an S3 bucket

Writing to S3's Intelligent storage class, or infrequent access tier, can reduce long-term costs. Intelligent storage automatically routes data to cheaper S3 tiers when it is not searched or touched for certain periods of time. Consult the AWS Cost Estimator to calculate the differences. Old data can also be rolled off to a much cheaper option such as Glacier Instant Retrieval or Glacier Deep Archive. Below are sample cost comparisons:


Standard & Infrequent access assumptions
Intelligent tiering assumptions
Cost comparisons
Auto-roll to infrequent access = less costs for Intelligent-tier
Storage
Requests & data retrievals
Data transfer
Management & analytics
Replication
S3 Object Lambda

60 TB per month

PUT, COPY, POST, LIST requests to S3 Standard
30000000

GET, SELECT, and all other requests from S3 Standard
5000000

Percentage of Storage in INT-Frequent Access Tier
Assumes no range GETs and that objects are only retrieved once
30 %

Percentage of Storage in INT-Infrequent Access Tier (% of storage that hasn't been accessed in the last 30 days)
Assumes no range GETs and that objects are only retrieved once
70 %

Percentage of Storage in INT-Archive Instant Access Tier (% of storage that hasn't been accessed in the last 90 days)
Assumes no range GETs and that objects are only retrieved once.
0 %

Percentage of Storage in INT-Archive Access Tier (% of storage that hasn't been accessed for a minimum of 90 days)
Assumes no range GETs and that objects are only retrieved once.
0 %

Percentage of Storage in INT-Deep Archive Access Tier (% of storage that hasn't been accessed for a minimum of 180 days)
Assumes no range GETs and that objects are only retrieved once.
%

Monitoring and Automation (objects per month)
Enter objects per month

S3 Standard estimate
Total monthly cost: 1,554.88 USD

S3 Intelligent - Tiering estimate
Total monthly cost: 1,113.54 USD

S3 Standard - Infrequent Access estimate
Total monthly cost: 1,073.00 USD

* S3 Intelligent-Tiering standard and bulk data retrieval and restore requests are free of charge for all five access tiers: Frequent, Infrequent, Archive Instant, Archive, and Deep Archive access tiers. Subsequent restore requests called on objects already being restored will be billed as a GET request. Expedited retrievals are available for the S3 Intelligent-Tiering Archive Access Tier and are charged at the Expedited request and retrieval rate.

Copyright © 2021 Cribl, Inc. All Rights Reserved.

AWS Cost Comparison

Cribl Stream Collectors and Replay features don't support S3 Glacier or S3 Deep Glacier due to long retrieval times for those storage classes.

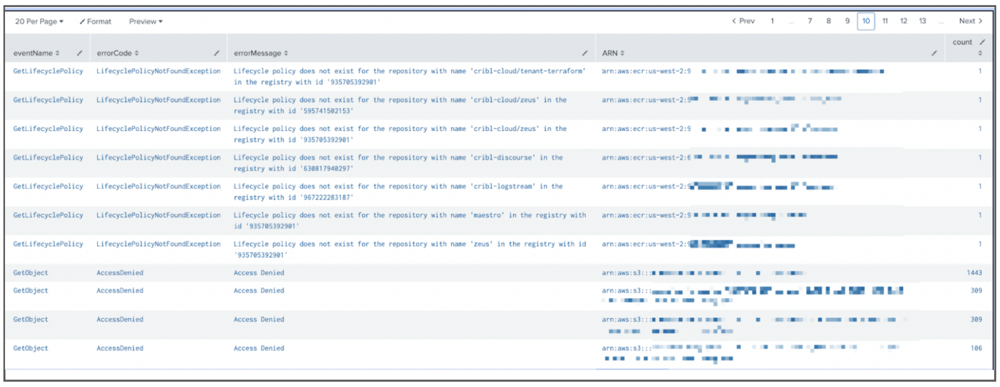
Collectors and Replay features do support S3 Glacier Instant Retrieval when you're using the S3 Intelligent-Tiering storage class.

Better Practice 8: Monitor Amazon S3 Buckets

AWS may limit S3 access if cumulative API writes are greater than 3,500 writes/second, or cumulative API reads are greater than 5,500 reads/second for a prefix (a.k.a., directory within S3). While it would be impossible to track every single AWS prefix's API requests, Cloudtrail offers "rate exceeded" events if you are ever over the API request limits.

Here's a Splunk search to report on all Cloudtrail events for S3. Filter or look for rate exceeded within the `errorMessage` field:

```
index=aws sourcetype=aws:cloudtrail errorCode=* "resources{}.ARN"=* | rename
"resources{}.ARN" as ARN | stats count by eventName errorCode errorMessage ARN
```



eventName	errorCode	errorMessage	ARN	count
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'cribl-cloud/tenant-terraform' in the registry with id '935705392901'	arn:aws:ecr:us-west-2:9[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'cribl-cloud/zeus' in the registry with id '595741502153'	arn:aws:ecr:us-west-2:5[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'cribl-cloud/zeus' in the registry with id '935705392901'	arn:aws:ecr:us-west-2:9[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'cribl-discourse' in the registry with id '630817940297'	arn:aws:ecr:us-west-2:6[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'cribl-logstream' in the registry with id '967222283187'	arn:aws:ecr:us-west-2:9[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'maestro' in the registry with id '935705392901'	arn:aws:ecr:us-west-2:9[redacted]	1
GetLifecyclePolicy	LifecyclePolicyNotFoundException	Lifecycle policy does not exist for the repository with name 'zeus' in the registry with id '935705392901'	arn:aws:ecr:us-west-2:9[redacted]	1
GetObject	AccessDenied	Access Denied	arn:aws:s3:::[redacted]	1443
GetObject	AccessDenied	Access Denied	arn:aws:s3:::[redacted]	309
GetObject	AccessDenied	Access Denied	arn:aws:s3:::[redacted]	309
GetObject	AccessDenied	Access Denied	arn:aws:s3:::[redacted]	106

Cloudtrail Logs Search

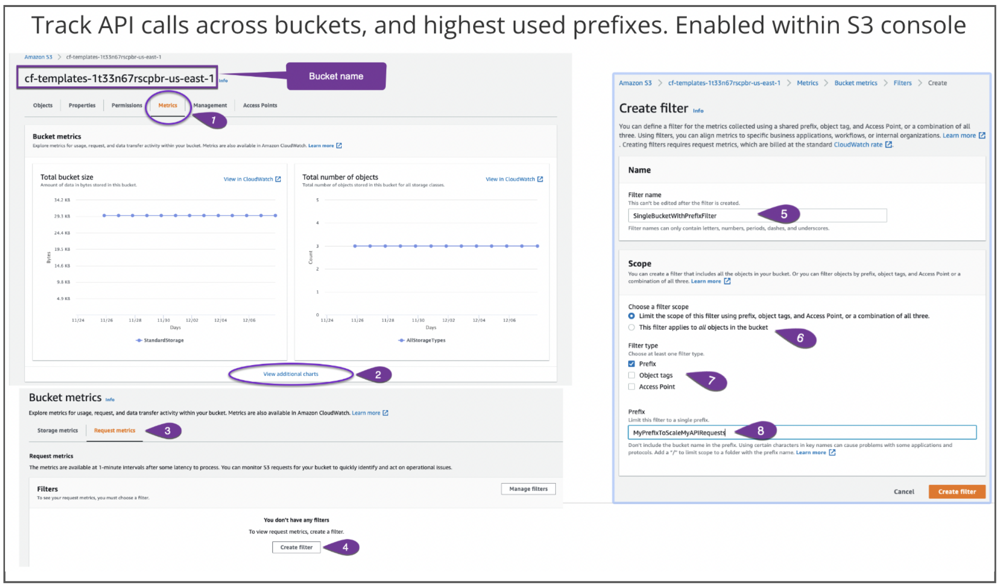
If there are `rate exceeded` messages from writing to Amazon S3, then a new strategy for partition and filename prefix expressions might be warranted. You will want to create fewer files, so consider one or more of the following options:

- Increasing timeouts.
- Eliminating file name expression prefixes.
- Updating partitioning expressions.

If there are rate exceeded messages on reading from Amazon S3, then you should likely apply better filtering. This might also warrant a new strategy for how files are written to S3.

For trending API calls, you'll need to focus on a specific prefix. This might be useful for specific AWS prefixes, as there will be hundreds of thousands or millions. API calls can be tracked via AWS Cloudwatch, but first need to be enabled for tracking within the S3 bucket. See AWS' Creating a Metrics Configuration that Filters by Prefix, Object Tag, or Access Point topic.

The screenshot below illustrates the locations to enable tracking API calls for a specific prefix:


Track API calls across buckets, and highest used prefixes. Enabled within S3 console

Amazon S3 > cf-templates-1t33n67rscpbr-us-east-1

cf-templates-1t33n67rscpbr-us-east-1 Info

Bucket metrics

Explore metrics for usage, request, and data transfer activity within your bucket. Metrics are also available in Amazon CloudWatch. Learn more

Total bucket size — Amount of data in bytes stored in this bucket.

Total number of objects — Total number of objects stored in this bucket for all storage classes.

Bucket metrics Info

Explore metrics for usage, request, and data transfer activity within your bucket. Metrics are also available in Amazon CloudWatch. Learn more

Request metrics

The metrics are available at 1-minute intervals after some latency to process. You can monitor S3 requests for your bucket to quickly identify and act on operational issues.

Filters

To see your request metrics, you must choose a filter.

You don't have any filters

To view request metrics, create a filter.

Amazon S3 > cf-templates-1t33n67rscpbr-us-east-1 > Metrics > Bucket metrics > Filters > Create

Create filter Info

You can define a filter for the metrics collected using a shared prefix, object tag, and Access Point, or a combination of all three. Using filters, you can align metrics to specific business applications, workflows, or internal organizations. Learn more . Creating filters requires request metrics, which are billed at the standard CloudWatch rate.

Name

Filter name: SingleBucketWithPrefixFilter

This can't be edited after the filter is created.

Filter names can only contain letters, numbers, periods, dashes, and underscores.

Scope

You can create a filter that includes all the objects in your bucket. Or you can filter objects by prefix, object tags, and Access Point or a combination of all three. Learn more

Choose a filter scope
- ● Limit the scope of this filter using prefix, object tags, and Access Point, or a combination of all three.
- ○ This filter applies to *all* objects in the bucket

Filter type — Choose at least one filter type.
- ☑ Prefix
- ☐ Object tags
- ☐ Access Point

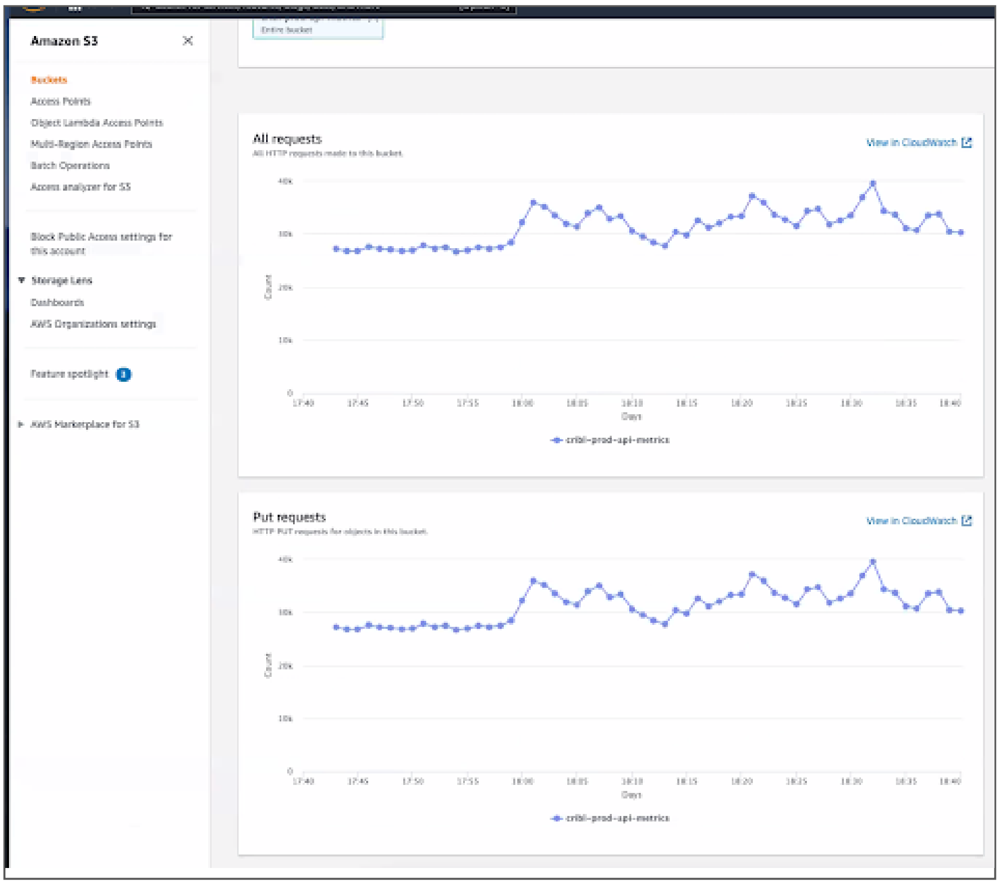
Prefix — Limit this filter to a single prefix.

MyPrefixToScaleMyAPIRequests

Don't include the bucket name in the prefix. Using certain characters in key names can cause problems with some applications and protocols. Add a "/" to limit scope to a folder with the prefix name. Learn more

Enable Monitoring in S3

Here's a Cloudwatch chart trending all requests, and write requests, for an S3 prefix. This is granular to 1 minute. Because the API limits are per second, some extrapolations are necessary to estimate per-second values.



Cloudwatch Chart

Consider both the write and read behavior when designing your partitioning expressions and file name prefix expressions.

Better Practice 9: Monitor Worker Node Performance, and Enable Notifications

Aside from the standard CPU load, disk space, memory usage metrics, consider tracking some additional metrics. If the Worker Nodes are deployed as EC2 instances, track the Cloudwatch metrics for the EC2 instances and associated disks. Cribl Support will ask for these metrics to track Amazon S3 issues:

- `BurstBalance`
- `VolumeReadBytes`
- `VolumeWriteBytes`
- `VolumeReadOps`
- `VolumeWriteOps`
- `VolumeQueueLength`

On the Linux host, the following command allows you to track the number of open files. This would be another useful metric to track over time:

```
lsof | grep <common_upperlevel_staging_dir> | wc -l
```

You can also configure Notifications in Cribl Stream when the S3 Destination is unhealthy or triggering backpressure. Once the S3 Destination is enabled, and a Notification target is set, you need to:

- Edit Cribl Stream's S3 Destination to configure Notifications.
- Add Notifications for different conditions.

Better Practice 10: Dedicate a Separate Worker Group for Replays

Execute Replays from a dedicated Worker Group. Streaming real-time data needs dedicated Worker Groups. You do not want CPU or memory conflicts on production systems that are streaming mission-critical, real-time machine data. Kubernetes or AWS spot instances are good options for on-demand Worker Processes that are needed only for infrequent replays.

Bonus Better Practice: Tune Amazon S3 Source Settings

For Cribl Stream's SQS-based Amazon S3 Source (as opposed to our S3 Collector), a few settings can speed up processing of new files added to an S3 bucket:

1. Under **General Settings**, apply a **Filename filter** if you want to pull only a subset of files from the S3 filter. The value is a regular expression (e.g., *suricata*)
2. Under **Advanced Settings**, set **Max messages** as high as 10 to fetch more SQS messages from the queue at once.
3. Under **Advanced Settings**, set the **Num receivers** to vertically scale the number of simultaneous processes pulling SQS messages and downloading S3 files. For S3 buckets with centralized Cloudtrail logs, or VPC flows across multiple accounts, raise the setting to 5 or 10.

Improving Partitioning Expressions

Creating partitioning expressions is an art. To get a sense of the possibilities and risks, let's consider both positive and negative examples.

Partitioning Expression 1: The Good

```
${C.Time.strftime(_time ? _time : Date.now()/ 1000, '%Y/%m/%d/%H/%M')}
```

Key Questions	Answers
Potential cardinality every *minute*?	1
How many open files if there are 30 Worker Processes?	30 open files
How many files in the S3 prefix, if the timeout is the default 60 seconds (assuming no file size limit)?	30 * 1 = 30 files
What if there are 10 Nodes in the Worker Group? How many files in each S3 prefix?	30 * 10 = 300 files
What happens when a replay fetches data for this hour?	Under API limit of 5,500 reads/sec

Result

- A good partitioning expression.

Partitioning Expression 2: The Ambiguous

```
${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')}
/${index ? index : 'no_index'}
/${host ? host : 'no_host'}
/${sourcetype ? sourcetype : 'no_sourcetype'}
```

Key Questions	Answers
Potential cardinality every 3 minutes?	3,000
How many active subdirectories in the staging dir?	3,000
What is the excess cardinality?	3,000 - 2,000 = 1,000
What are maximum open files if there are 14 Worker Processes on a Worker ?	14 * 2,000 = 28,000 open files
What happens when maximum open files of 2,000 is reached for any one Worker Process? What happens if it happens too frequently?	All files closed & written to S3 at once = 2,000 writes. Potential for backpressure if Cribl Stream takes long enough to write the files.
What's the maximum number of files in any S3 prefix?	14 * 10 Worker Process * (60min/3min) = 2800 files max in any S3 prefix

Additional assumptions:

- Cardinality of 3,000 every 3 mins.
- File write timeout of 3 minutes.
- Max file size never reached.
- 10 Worker Process in group.

Result

- Proceed with caution.

Solution

- Track the occurrences of `max open files reached` messages
- If the `max open files reached` message occurs too frequently, consider reducing the cardinality. You might achieve this by lowering the timeout period to 2 minutes or 1 minute.

Partitioning Expression 3: The Bad

```
${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d/%H')}
```

Key Questions	Answers
Potential cardinality every hour?	1
How many open files if there are 30 Worker Processes?	30 open files.
How many files in the S3 prefix, if the timeout is the default 60 seconds (assuming no file size limit)?	(30 * 60) = 1800 files
What if there are 10 Nodes in the Worker Group? How many files in each S3 prefix?	(1,800 * 10) = 18,000 files
What happens when a replay fetches data for this hour?	Could exceeed the API limit of 5,500 reads/sec.

Result

- A bad partitioning expression.

Solution

- Change the timeout to 1800 seconds to allow files to grow in size.

Partitioning Expression 4: The Terrible

```
${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d')}
/${index ? index : 'no_index'}
/${host ? host : 'no_host'}
/${sourcetype ? sourcetype : 'no_sourcetype'}
```

Filename Prefix Expression:

```
${C.Time.strftime(_time ? _time : Date.now() / 1000, '%H%M')}
```

Key Questions	Answers
What are max open files on any one Worker Process?	Answer: `2,000`, because that is the lower of: `[(500 directories * 5 filename prefix files = 2,500 open files per process]` and `[2,000 open files]`

Key Questions	Answers
How many active subdirectories in the staging dir?	Around 400
What happens when max open files of 2,000 is reached for any one Worker Process?	All files closed and written to S3 at once (2,000 writes at once). Happens every 4 minutes. There's potential for backpressure if Cribl Stream takes long enough to write the files.
What's the max number of files in any S3 prefix?	14 processes * 1440 mins * 10 Worker Process = 202K files! API issues on read are likely.

Additional Assumptions

- Reduced cardinality of 200 every 2 minutes, and 500 every 5 minutes.
- Introduction of a file prefix partition.
- Max open files within Cribl Destination set to 2,000.
- Max file open time of 5 minutes.
- Assume max file size is never reached.
- 10 Worker Processes in Group.
- 14 Worker Processes/Worker.

Result

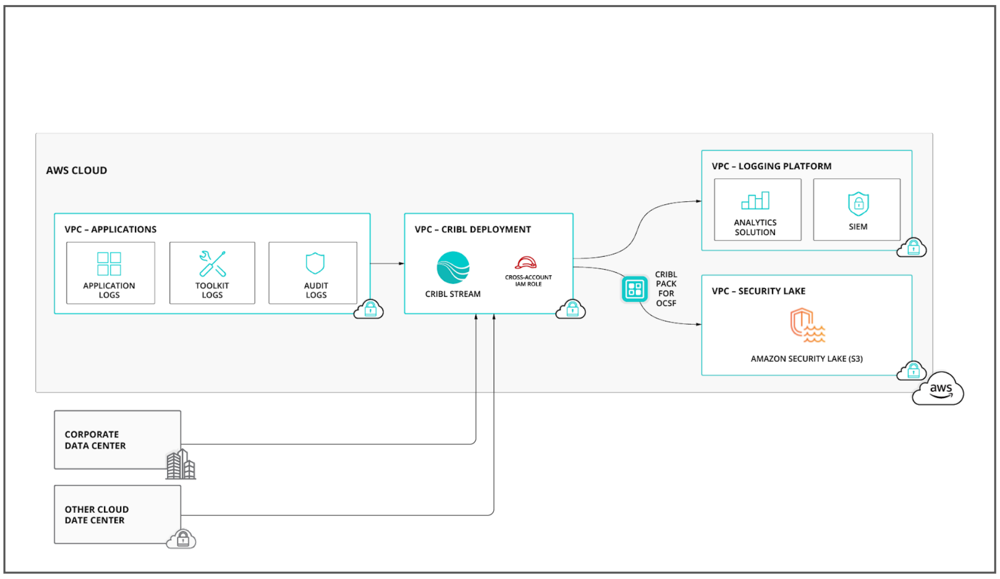
- A terrible partitioning expression.

Solution

- Don't use the file prefix expression; consider expanding the partitioning expression instead.
- Reduce timeout to 2 minutes, as cardinality will be lower over a 2-minute span.

17.6.1.2. Amazon Security Lake Integration

Cribl Stream 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 Stream with Amazon Security Lake

This guide walks you through an example where Cribl Stream, running in Cribl.Cloud, sends Palo Alto Networks logs to Amazon Security Lake. The workflow starts with a Cribl Stream Syslog Source, which ingests the Palo Alto data and routes it to a Cribl Stream 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 Stream 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 Stream

Complete the procedures in this section before doing anything else in Cribl Stream.

First, find and note the Amazon Resource Name (ARN) that enables AWS to locate your Cribl Stream 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 Stream 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 **Worker Groups** page.
3. Click the name of the Worker Group whose Worker Nodes you want to send data to Amazon Security Lake. (For this example, we'll use the `default` Worker Group.) 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 Stream 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 Stream 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 Stream Amazon Security Lake Destination.

Please post any questions you have about these procedures to the `#packs` channel of Cribl Community Slack.

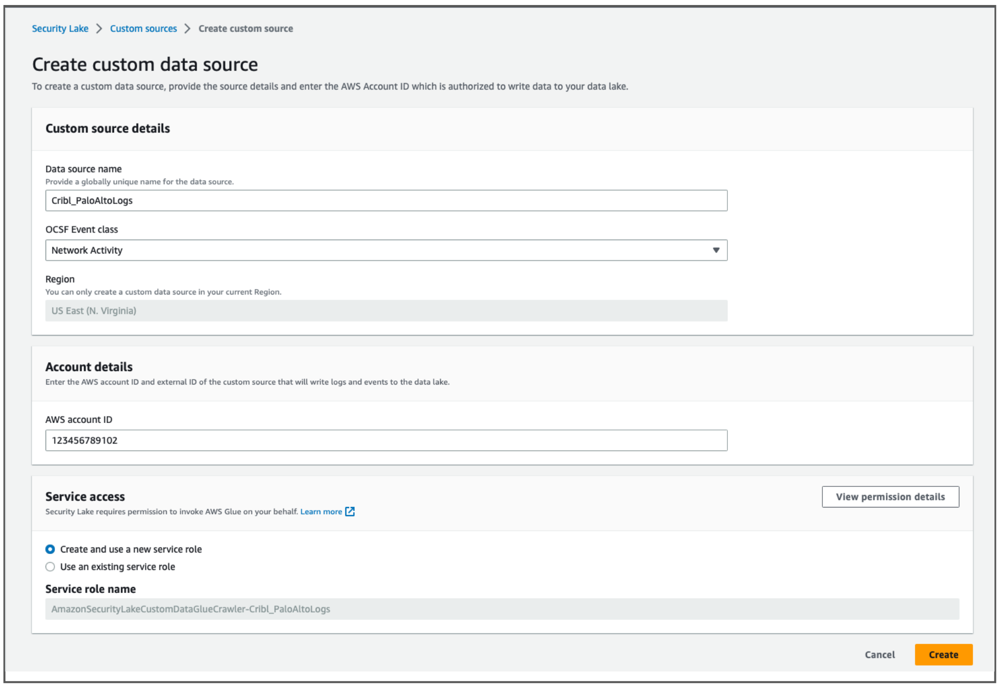
Creating an Amazon Security Lake Custom Source

Because Cribl is not a native Amazon service, you'll configure what Amazon calls a **Security Lake custom source**, and the Cribl Stream 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.


Security Lake > Custom sources > Create custom source

Create custom data source

To create a custom data source, provide the source details and enter the AWS Account ID which is authorized to write data to your data lake.

Custom source details

Data source name
Provide a globally unique name for the data source.
Cribl_PaloAltoLogs

OCSF Event class
Network Activity

Region
You can only create a custom data source in your current Region.
US East (N. Virginia)

Account details
Enter the AWS account ID and external ID of the custom source that will write logs and events to the data lake.

AWS account ID
123456789102

Service access
Security Lake requires permission to invoke AWS Glue on your behalf. Learn more

● Create and use a new service role
○ Use an existing service role

Service role name
AmazonSecurityLakeCustomDataGlueCrawler-Cribl_PaloAltoLogs

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 Stream 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 Stream, create a new Amazon Security Lake Destination in the Routing UI as described in the Destination topic. In the **New Destination** modal, configure the settings specified below. For all settings **not** mentioned in the following notes, you can keep the defaults.

General Settings

S3 Bucket name: The S3 bucket name you noted when setting Up Amazon Security Lake.

Region: Select the region where the S3 bucket is located.

Optional Settings

File name prefix expression: Keep the default (`CriblOut`) if it satisfies your requirements; otherwise, edit to suit your needs.

Authentication

Authentication method must be set to `Auto` (the default).

Assume Role

Cribl strongly recommends using the **Auto** authentication method in conjunction with the **Assume Role** settings as described below. Both Cribl and Amazon discourage the use of other approaches.

When using Assume Role to access resources in a different region than Cribl Stream, you can target the AWS Security Token Service (STS) endpoint specific to that region by using the `CRIBL_AWS_STS_REGION` environment variable on your Worker Node. Setting an invalid region results in a fallback to the global STS endpoint.

AssumeRole ARN: This is the **Provider role ARN** you copied after creating your custom source in Amazon Security Lake.

External ID: Enter the ID that you specified when creating your Amazon Security Lake custom source.

Processing Settings

Post-Processing

Pipeline: From the drop-down, select the Pack you installed earlier:

```
PACK Cribl-OCSF_Post_Processor (OCSF Post Processor)
```

System fields: Remove any fields specified here.

Parquet Settings

Parquet schema: From the drop-down select `OCSF 4001 Network Activity`, since OCSF Event Class 4001 describes the events coming from Palo Alto Networks in this example.

At this point, you can click **Save**; the defaults for **Advanced Settings** should work fine for this example. Then, **Commit** and **Deploy**.

You must create one Cribl Stream 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 Stream 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 Stream Syslog Source to ingest Palo Alto Networks system logs, using the QuickConnect UI. (The reference below shows what Cribl Stream 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 Stream.

Troubleshooting and Testing

Good things to double-check:

- Are you sending the events to your Amazon Security Lake in Parquet format?
- Are your permissions set properly for your IAM role to write to the S3 bucket?

If the answer to either question is "No," your data will not make it into Amazon Security Lake.

To send sample events from the GitHub repo, using netcat:

1. Download the `oYLbFU.json` sample file.
2. Run the following command, replacing `<cribl_org_id>` with your Cribl.Cloud Organization ID:

```
cat oYLbFU.json | nc default.main-<cribl_org_id>.cribl.cloud 10070
```

With a Cribl.Cloud Enterprise plan, generalize the above URL's `default.main` substring to `<group-name>.main` when sending to other Worker Groups.

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 Stream. This holds true as long as the OCSF Event Classes you want to work with are among those supported by Cribl Stream. 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`

Data Source	OCSF Event Classes Handled	Cribl Source	Cribl Pack Display Name	OCSF 3004 Entity Management ema(s)
Azure NSG Flow Logs	4001	Azure Blob Storage Source, run as scheduled jobs, not as a Pull Source	Azure NSG Flow Logs	`OCSF 4001 Network Activity`
Cisco ASA	4001	Syslog	Cisco ASA	`OCSF 4001 Network Activity`
Cisco FTD	4001	Syslog	Cisco FTD	`OCSF 4001 Network Activity`
CrowdStrike Account Change	3001	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 3001 Account Change`
CrowdStrike Authentication	3002	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 3002 Authentication`
CrowdStrike Network Activity	4001	CrowdStrike FDR	Crowdstrike FDR Pack	`OCSF 4001 Network Activity`
GCP Audit Logs Account Activity	3001	Google Cloud Pub/Sub	Google Cloud Audit Logs	`OCSF 3001 Account Change`
Palo Alto Networks (PAN) Threat and Traffic	4001	Syslog	OCSF Post Processor Pack	`OCSF 4001 Network Activity`

SentinelOne	1001, 2001, 3002, 4001	Amazon S3	SentinelOne Cloud Funnel	`OCSF 1001 File System Activity`
				
				`OCSF 2001 Security Finding`
				
				`OCSF 3002 Authentication`
				
				`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`

17.6.1.3. Collecting Logs From Amazon ECS Containers

When using the Amazon Elastic Container Service (ECS), Cribl Stream can eliminate the additional expense of CloudWatch Logs by selecting an alternative Docker logging driver.

By default, the `awslogs` driver will forward all container logs to CloudWatch Logs. This requires an additional Amazon service, such as Kinesis Firehose or Lambda, to export logs to the desired destination.

But by using Docker's `splunk` logging driver, you can forward the ECS container logs directly to a Cribl Stream Worker Group, bypassing CloudWatch Logs.

For example, when using Cribl.Cloud, you can forward logs to your Organization's Splunk HEC endpoint. This endpoint information can be found in your Cloud Organization's Data Onboarding configuration.

Example Task Definition

Below is a sample JSON task definition with the required `logConfiguration` section configured to forward Splunk HEC payloads to Cribl Stream:

```
{
    "containerDefinitions": [
        {
            "name": "my-example-container",
            "essential": true,
            "image": "public.ecr.aws/my/containerimage:latest",
            ...
            "logConfiguration": {
                "logDriver": "splunk",
                "options": {
                    "splunk-url": "https://main-default-<org>.cribl.cloud:8088",
                    "splunk-token": "176FCEBF-4CF5-4EDF-91BC-703796522D20",
                    ...
                }
            }
        }
    ],
    ...
}
```

Additional Details

For additional configuration options on the `splunk` logging driver, see Docker’s documentation.

For additional details on the `LogConfiguration` section, see the Amazon ECS documentation.

17.6.2. Azure

17.6.2.1. Preparing The Azure Workspace

Currently, Cribl Stream supports sending data only to custom tables and certain native tables. For details, see Azure's Supported Tables topic.

To send data to custom tables from Cribl Stream, you must update a DCR Template with Stream definitions. We've provided an example DCR Template file.

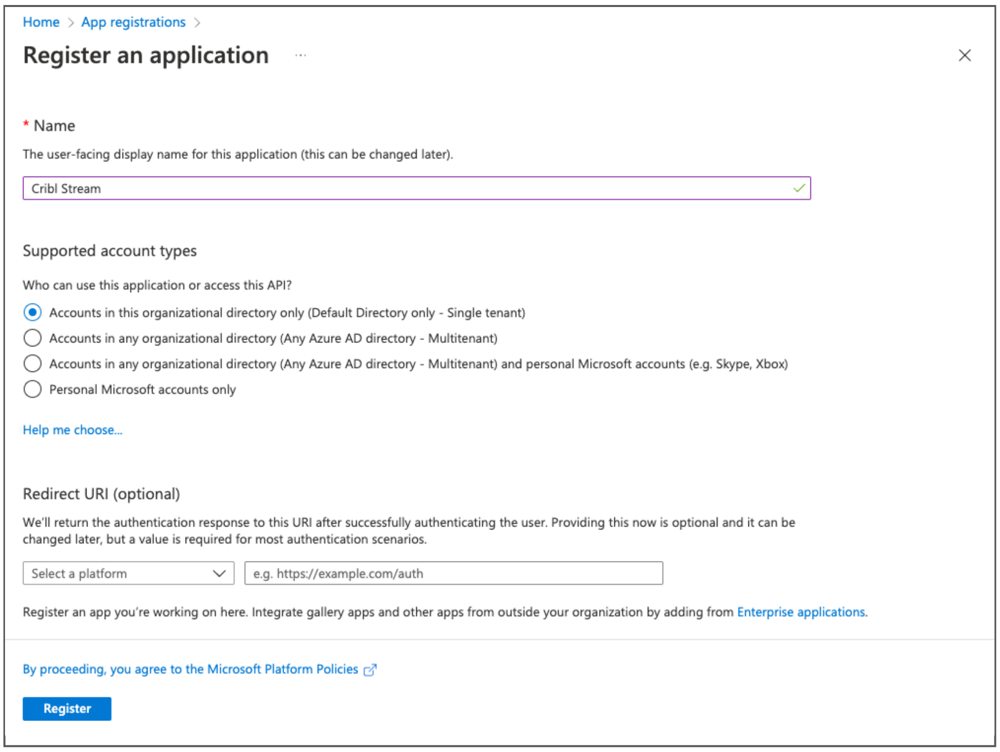
Prepare the Azure Workspace

First, you must prepare the Azure Workspace to receive data from Cribl Stream, by registering Cribl Stream as an authorized application with the Microsoft Identity Platform.

Use Copy buttons, where available, to copy complete values for later use without truncation.

Create Credentials for a New Azure Application

1. Log into the **Azure** portal as an `Administrator`.
2. Navigate to the portal's **App registration** section.
3. Click **New registration** to register an application.
4. Register a new application with the name `Cribl Stream`.


Home > App registrations >

Register an application

* Name

The user-facing display name for this application (this can be changed later).

Cribl Stream

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (Default Directory only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

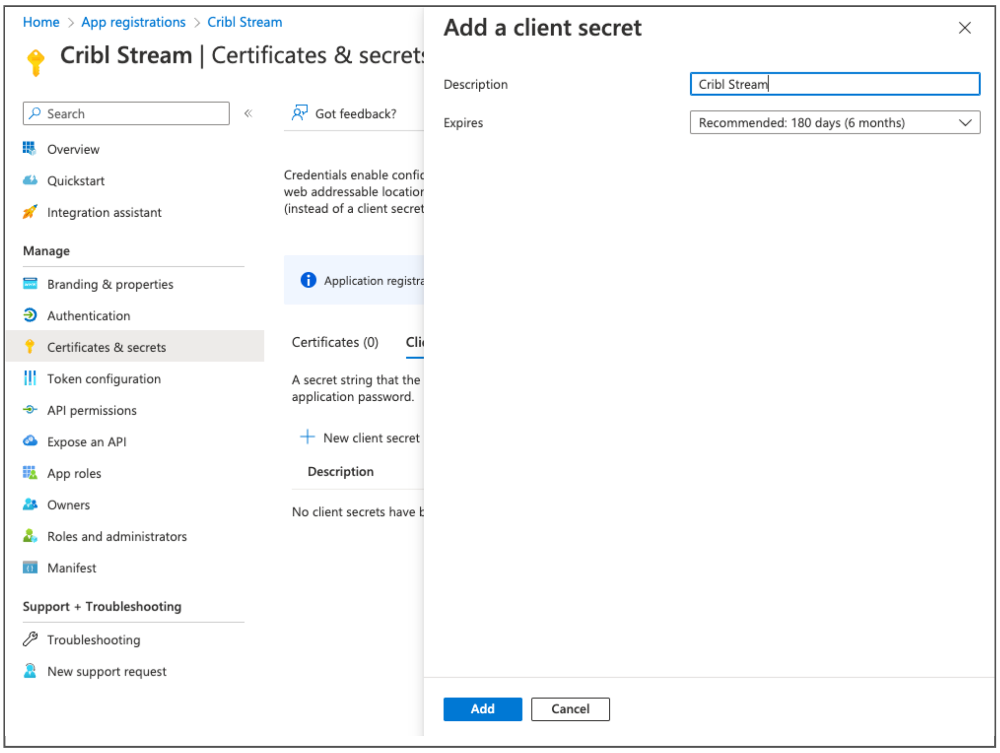
Select a platform | e.g. https://example.com/auth

Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

By proceeding, you agree to the Microsoft Platform Policies

Registering an Application

5. Click **Register**.
6. Store the **Application (client) ID** and **Directory (tenant) ID** for later steps.
7. In your newly registered application, select the **Certificates & secrets** tab.
8. Create a **New client secret**.
9. Store the secret **Value** for later steps.


Home > App registrations > Cribl Stream

Cribl Stream | Certificates & secrets

Credentials enable confic web addressable locatior (instead of a client secret

Application registra

Certificates (0) Cli

A secret string that the application password.

New client secret

Description

No client secrets have b

Add a client secret

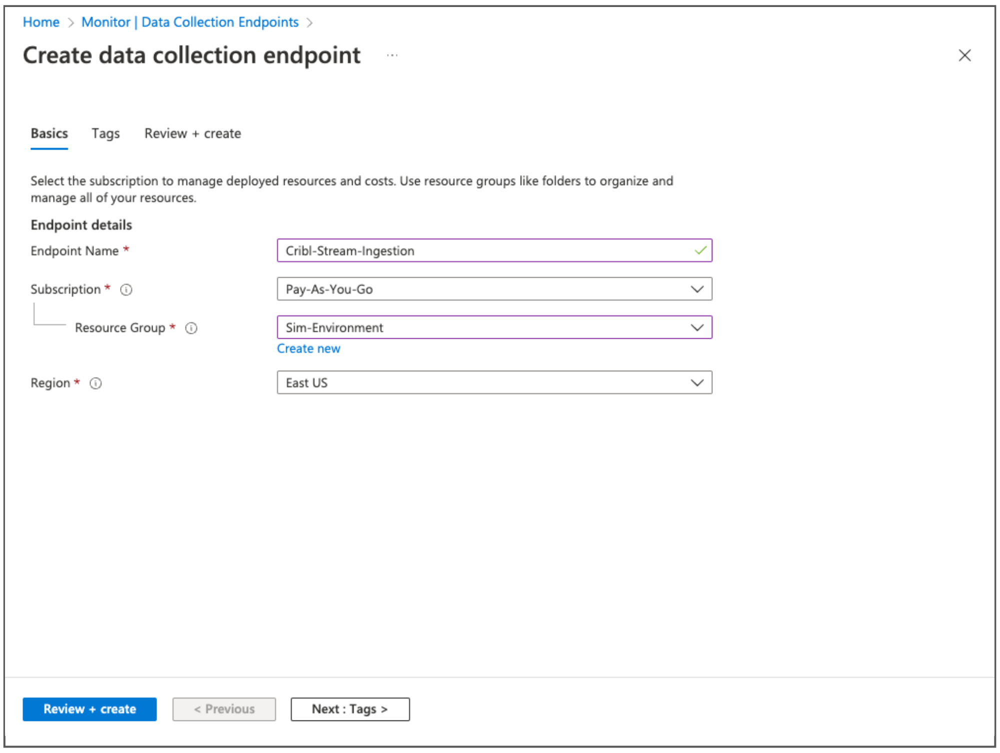
Description: Cribl Stream

Expires: Recommended: 180 days (6 months)

Creating a client secret

Create a Data Collection Endpoint

1. Navigate to the Azure portal's **Monitor** service.
2. Under **Settings**, select **Data Collection Endpoints**.
3. Select **Create** to add a new endpoint named `Cribl-Stream-Ingestion`.
4. Select the appropriate Subscription and Resource Group used by your organization.
5. Select **Review + create**, review your changes, then select **Create**.
6. In the list of **Data Collection Endpoints**, open the newly created endpoint. (You might need to refresh the page to see it.)


Home > Monitor | Data Collection Endpoints >
Create data collection endpoint
Basics
Tags
Review + create

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all of your resources.

Endpoint details

Endpoint Name * : Cribl-Stream-Ingestion

Subscription * : Pay-As-You-Go

Resource Group * : Sim-Environment

Create new

Region * : East US

Review + create
< Previous
Next : Tags >

Creating a Data Collection Endpoint

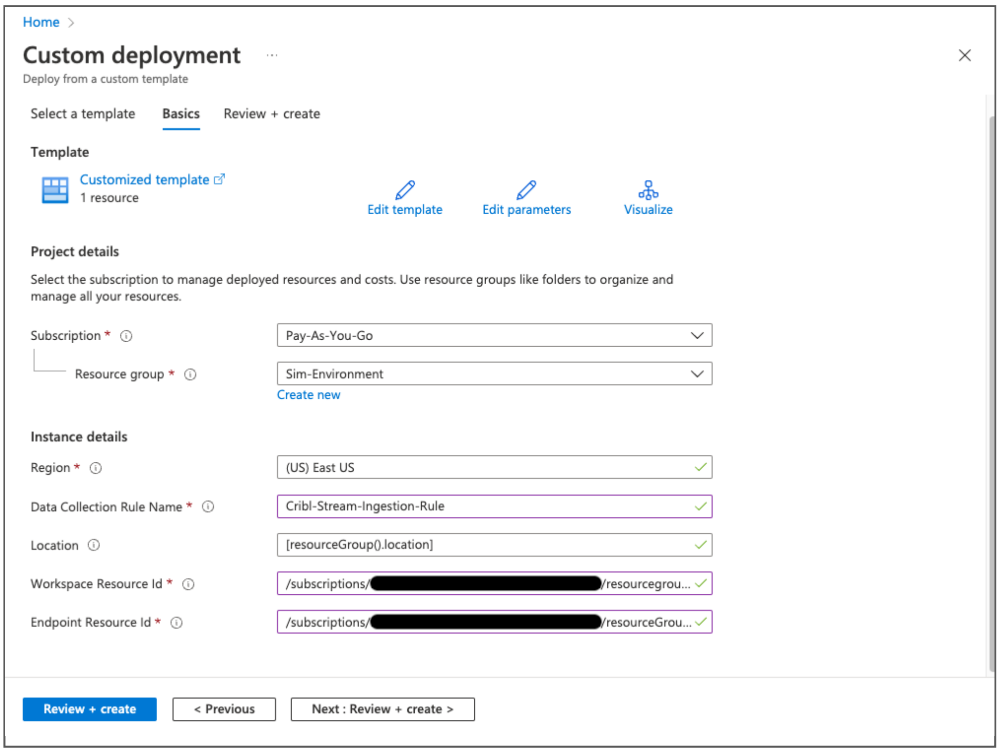
7. Navigate to the **Overview** page, and copy and store the **Logs Ingestion** URL.
8. Select **JSON view**, and store the value of the **Resource ID** (gray box at the top) for later use.

Find the Log Analytics Workspace Resource ID

1. Navigate to the Azure portal's **Log Analytics workspaces** service.
2. Select the workspace that will receive data.
3. From the **Overview** page, select **JSON view**, and store the value of the **Resource ID** (gray box at the top) for later use.

Create Data Collection Rule

1. Navigate to the Azure portal's **Deploy a custom template** service.
2. Select **Build your own template in the editor**.
3. Select **Load file** and upload the DCR Template.
4. Click **Save**.
5. Select the appropriate Subscription and Resource Group used by your organization.
6. Name the new Data Collection Rule `Cribl-Stream-Ingestion-Rule`.
7. Enter the Log Analytics **Workspace Resource ID**, and the Data Collection **Endpoint Resource ID**, that you stored in the previous steps.
8. To create the Data Collection Rule, click **Review + create**, followed by **Create**.



Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Pay-As-You-Go

Resource group * : Sim-Environment

Create new

Instance details

Region * : (US) East US

Data Collection Rule Name * : Cribl-Stream-Ingestion-Rule

Location : [resourceGroup().location]

Workspace Resource Id * : /subscriptions/[redacted]/resourcegrou...

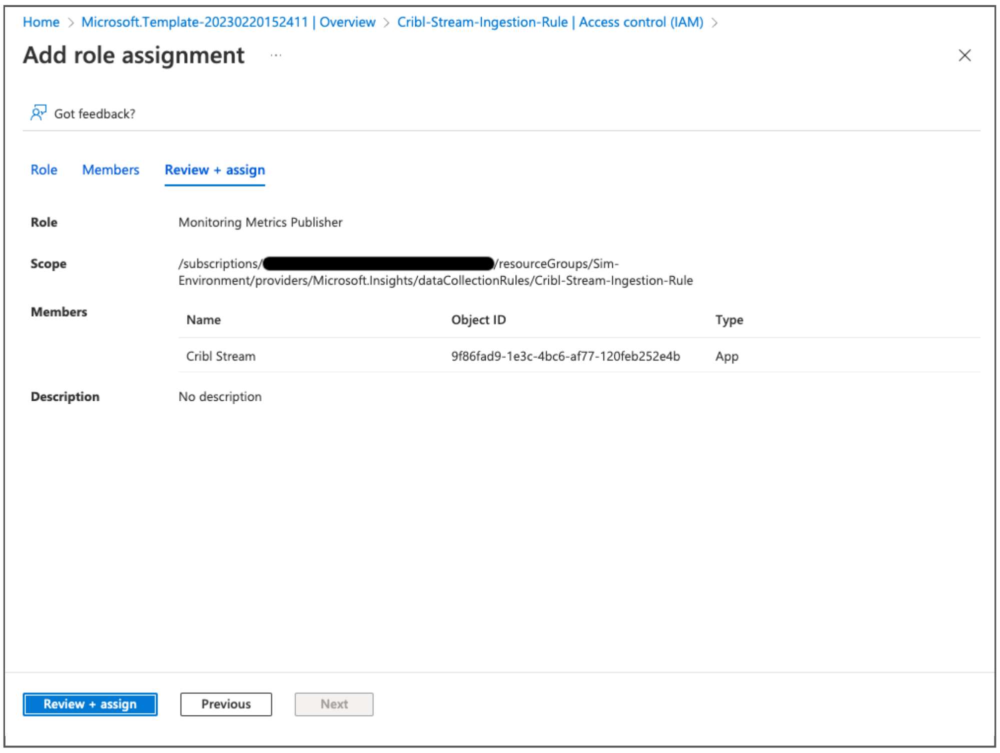
Endpoint Resource Id * : /subscriptions/[redacted]/resourceGrou...

Registering an Application

Add Role Assignment

1. Once the template has been deployed, click **Go to resource**.
2. From the Overview page, select the **JSON view**, and store the **immutableId** from the JSON body (without quotes) for later use.
3. Click the Data Collection Rule's **Access control (IAM)**.
4. Click **Add role assignment**.
5. Select the **Monitoring Metrics Publisher** role, then click **Next**.

6. Click **+ Select members**.
7. Search for and select `Cribl Stream` (the app you created earlier), then click **Select** to confirm the selection.
8. Click **Review + assign** to review changes.
9. Click **Review + assign** again to implement the permissions update.


Home > Microsoft.Template-20230220152411 | Overview > Cribl-Stream-Ingestion-Rule | Access control (IAM) >
Add role assignment
Got feedback?
Role
Members
Review + assign
Review + assign
Previous
Next

Role	Monitoring Metrics Publisher
Scope	/subscriptions/[redacted]/resourceGroups/Sim-Environment/providers/Microsoft.Insights/dataCollectionRules/Cribl-Stream-Ingestion-Rule
Members	
Description	No description

Name	Object ID	Type
Cribl Stream	9f86fad9-1e3c-4bc6-af77-120feb252e4b	App

Adding a Role Assignment

17.6.2.2. Microsoft Entra ID + OpenID Configuration

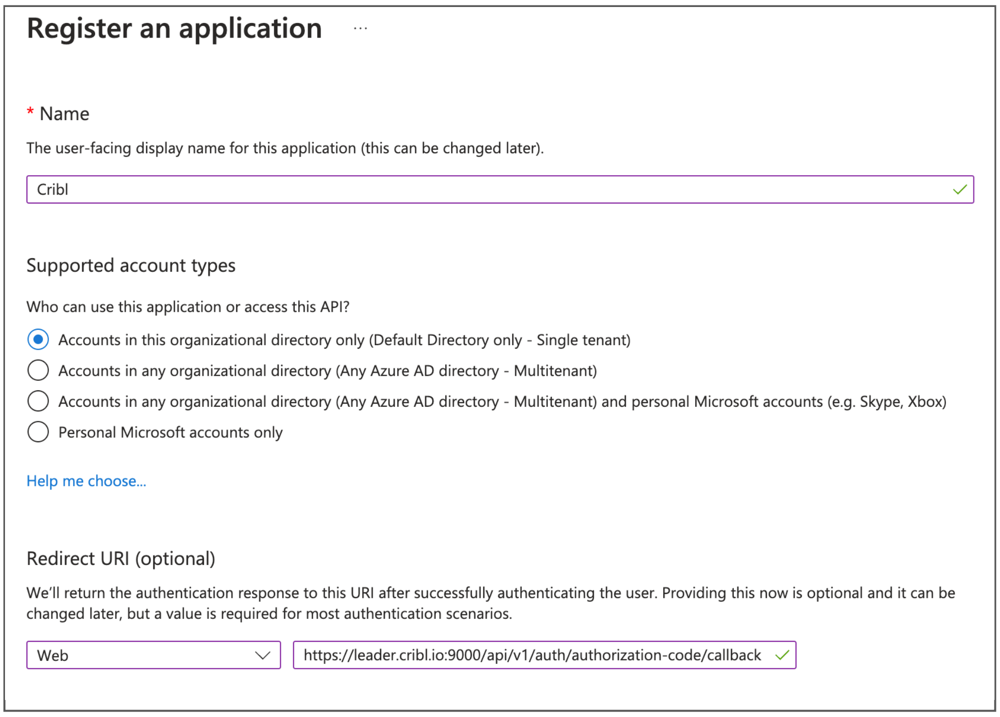
This page outlines how to integrate Azure Active Directory with Cribl Stream's SSO/OpenID Connect authentication. <!– should the 5 instances of Azure Active Directory on this page be replaced with Microsoft Entra ID? ->

Configure Microsoft Entra ID App

Start at the Azure portal to configure an OpenID Connect provider: https://portal.azure.com/.

Register Your Microsoft Entra ID App

1. Open the **Microsoft Entra ID** Service.
2. In the left nav's **Manage** section, select **App registrations**.
3. Add a new registration. For details, see Microsoft's Quickstart: Register an Application topic.
   In the example below, substitute the appropriate callback URL for your own Cribl Stream Leader instance: `https://leader.cribl.io:9000/api/v1/auth/authorization-code/callback`



Register an application

* Name

The user-facing display name for this application (this can be changed later).

Cribl ✓

Supported account types

Who can use this application or access this API?

- [✓ Accounts in this organizational directory only (Default Directory only - Single tenant)]
- Accounts in any organizational directory (Any Azure AD directory - Multitenant)
- Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

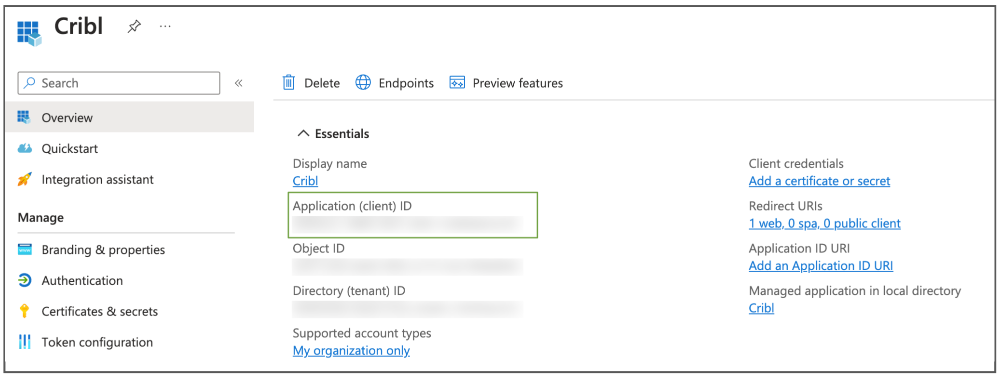
Web | https://leader.cribl.io:9000/api/v1/auth/authorization-code/callback ✓

Registering a Microsoft Entra ID app

Get the Microsoft Entra ID App's Basic Credentials

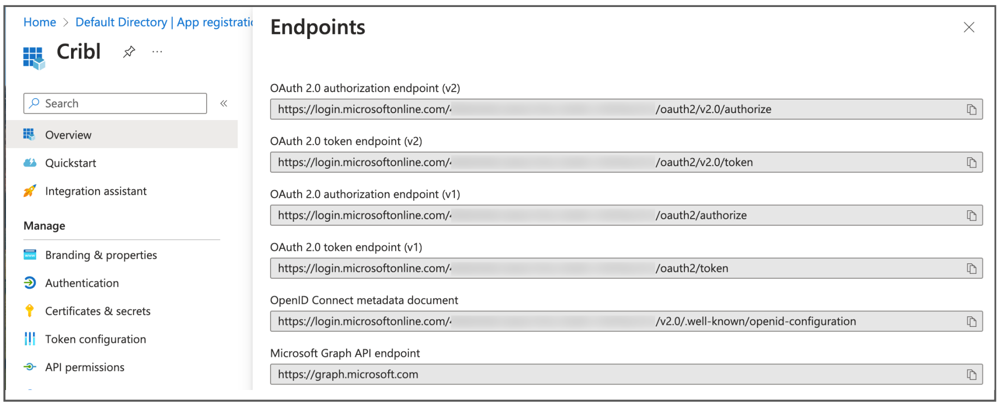
You'll need to copy and paste these credentials into Cribl Stream's **Authentication** page below.

1. You can find the OIDC Client ID on the new app's **Overview** page, as the **Application (client) ID**.



Finding the OIDC Client ID

2. Click the **Endpoints** button at the page top to display the OAuth endpoints. You can use either the v2 or the v1 endpoints.



OAuth 2.0 authorization endpoint (v2)
https://login.microsoftonline.com/ /oauth2/v2.0/authorize

OAuth 2.0 token endpoint (v2)
https://login.microsoftonline.com/ /oauth2/v2.0/token

OAuth 2.0 authorization endpoint (v1)
https://login.microsoftonline.com/ /oauth2/authorize

OAuth 2.0 token endpoint (v1)
https://login.microsoftonline.com/ /oauth2/token

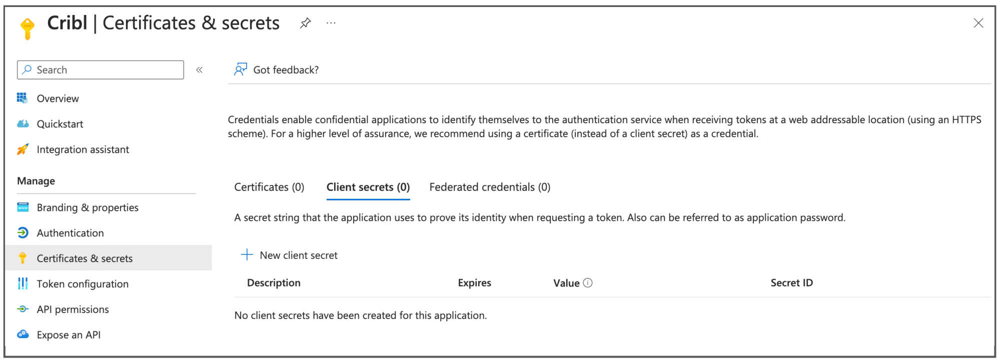
OpenID Connect metadata document
https://login.microsoftonline.com/ /v2.0/.well-known/openid-configuration

Microsoft Graph API endpoint
https://graph.microsoft.com

Copying OAuth 2 v2 endpoints

Create and Copy a Client Secret

1. To create a client secret: From the Azure portal's left nav, select **Certificate & secrets**. Then select **New client secret**.



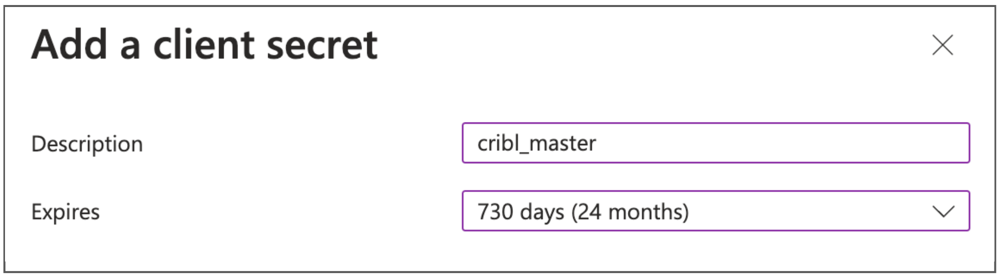
Credentials enable confidential applications to identify themselves to the authentication service when receiving tokens at a web addressable location (using an HTTPS scheme). For a higher level of assurance, we recommend using a certificate (instead of a client secret) as a credential.

A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

No client secrets have been created for this application.

Accessing client secrets

2. Add a new client secret with a descriptive name, and an expiration timeframe.

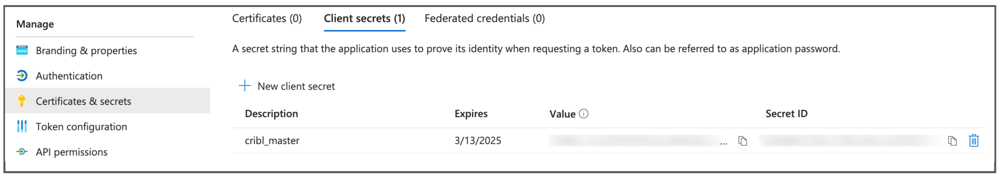


Add a client secret

Description	cribl_master
Expires	730 days (24 months)

Adding a client secret

3. Click **Add**.
4. Immediately copy the **Value** and **Secret ID** from the resulting page. You'll need to paste the **Value** into Cribl Stream's **Authentication** > **Client secret** field below.



A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	Secret ID
cribl_master	3/13/2025		

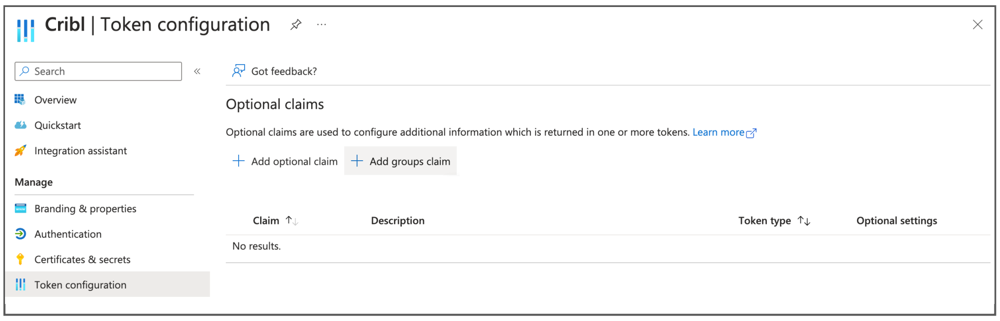
Copy that secret!

This is the only time the secret is shown! Make sure you copy it while it's visible. (If you missed your chance, you can start over by creating a new secret.)

Configure Token and Claims

Here, you'll add the groups claim to the OIDC ID token.

1. From the Azure portal's left nav, select **Token configuration**, then select **Add groups claim**.



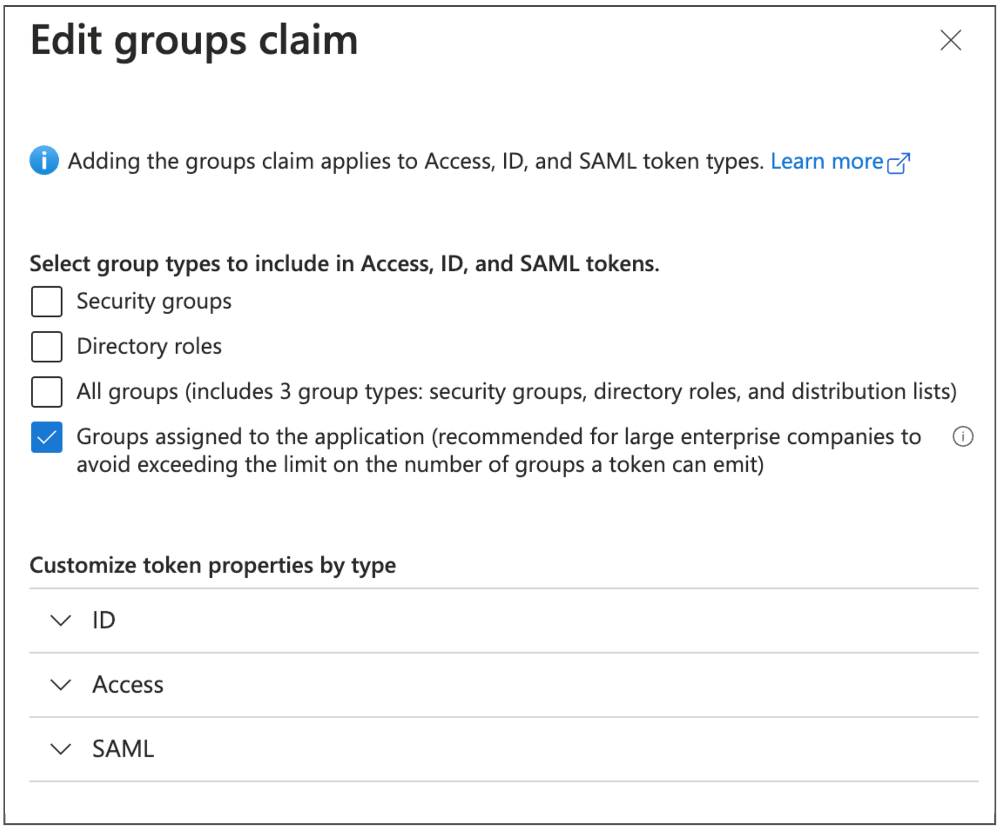
Optional claims

Optional claims are used to configure additional information which is returned in one or more tokens. Learn more

Claim	Description	Token type	Optional settings
No results.			

Configuring a token

2. Configure the groups claim as necessary, then click **Add**.


Edit groups claim

Adding the groups claim applies to Access, ID, and SAML token types. Learn more

Select group types to include in Access, ID, and SAML tokens.

- [ ] Security groups
- [ ] Directory roles
- [ ] All groups (includes 3 group types: security groups, directory roles, and distribution lists)
- [✓] Groups assigned to the application (recommended for large enterprise companies to avoid exceeding the limit on the number of groups a token can emit)

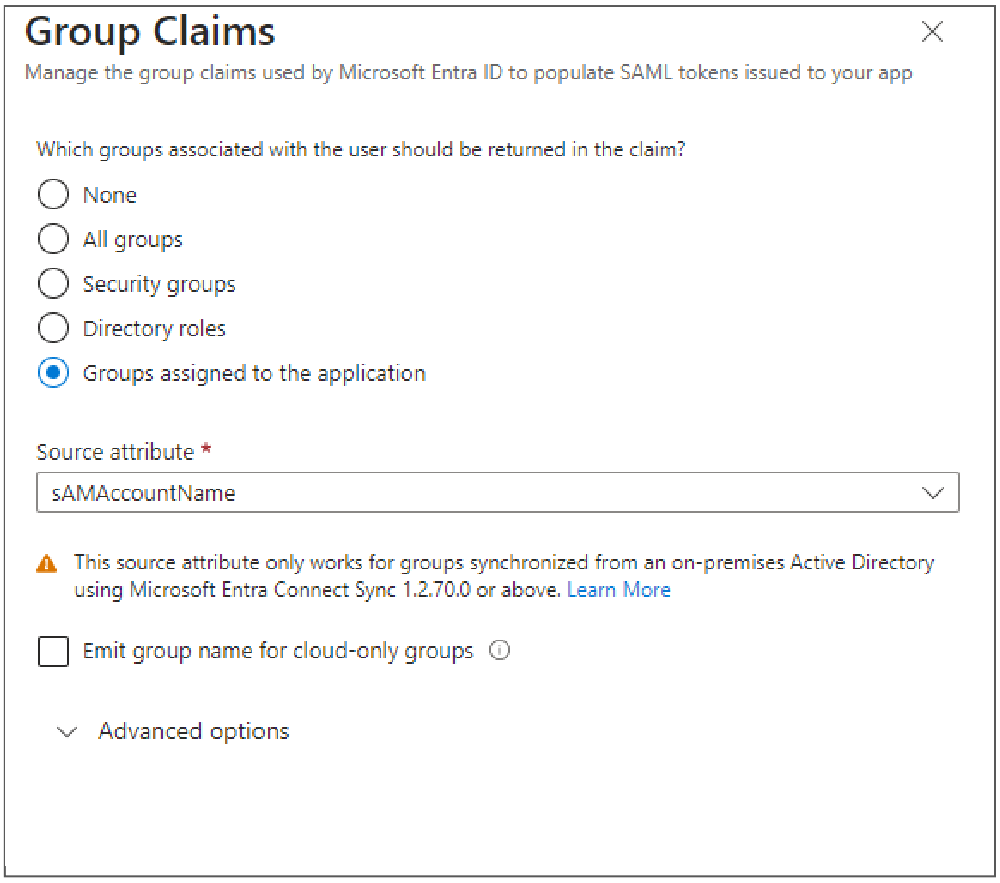
Customize token properties by type

- ID
- Access
- SAML

Editing the groups claim

Unless you synchronize Azure AD with your on-premises Active Directory, AD will return only GUIDs for your group names. If you've synchronized, you'll then be able to configure returning the `sAMAccountName` instead.

In the **Group claims** modal (shown below), check the **Emit group name for cloud-only groups** checkbox to ensure that `sAMAccountName` is added to the **Group** attribute.



Group Claims

Manage the group claims used by Microsoft Entra ID to populate SAML tokens issued to your app

Which groups associated with the user should be returned in the claim?

- None
- All groups
- Security groups
- Directory roles
- [✓ Groups assigned to the application]

Source attribute *

sAMAccountName

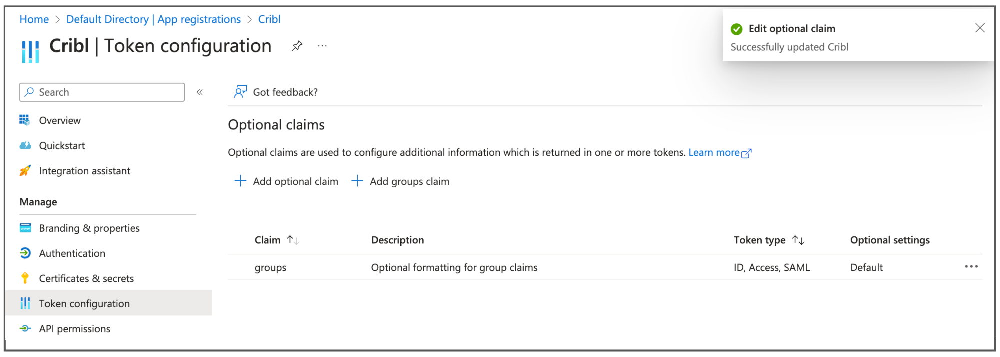
This source attribute only works for groups synchronized from an on-premises Active Directory using Microsoft Entra Connect Sync 1.2.70.0 or above. Learn More

- [ ] Emit group name for cloud-only groups

Advanced options

Editing the groups claim

3. Your token is now configured, and you're all done on the Azure side.



Home > Default Directory | App registrations > Cribl

Cribl | Token configuration

Edit optional claim
Successfully updated Cribl

Optional claims

Optional claims are used to configure additional information which is returned in one or more tokens. Learn more

Claim	Description	Token type	Optional settings
groups	Optional formatting for group claims	ID, Access, SAML	Default

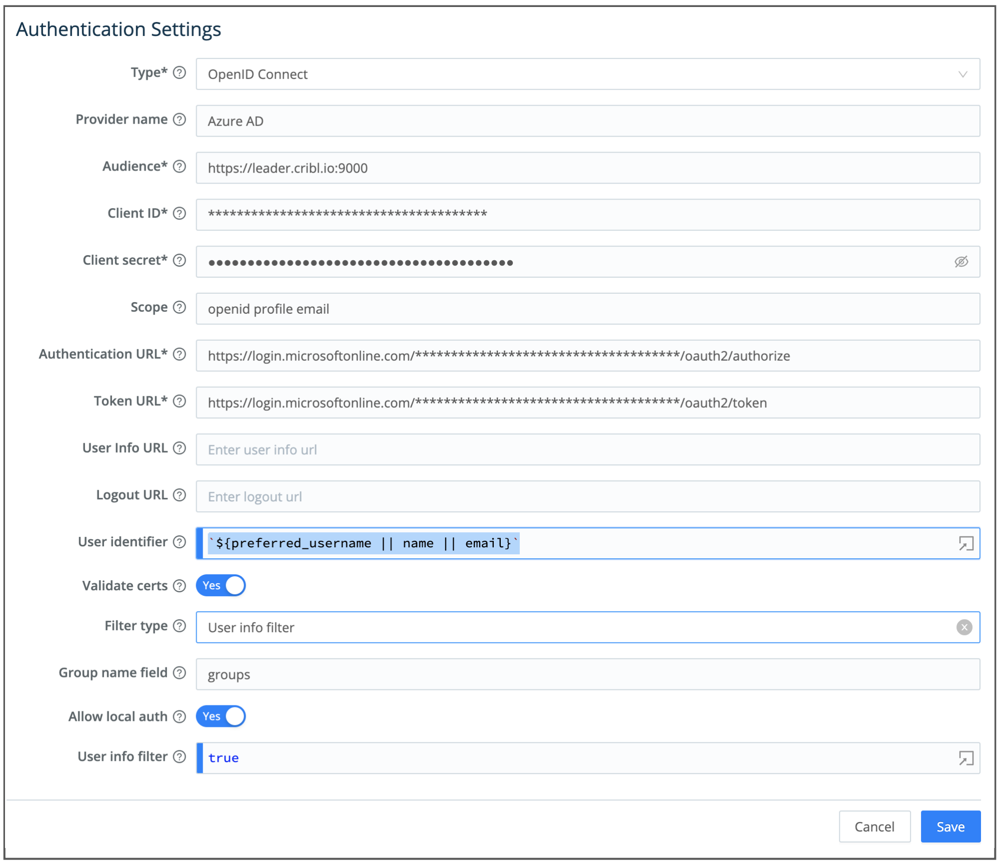
Microsoft Entra ID token configuration complete

Configure Cribl Stream Authentication

Switch to Cribl Stream, and navigate to its **Settings** > [**Global Settings** >] **Access Management** > **Authentication** page. Configure this as indicated below (with reactions):

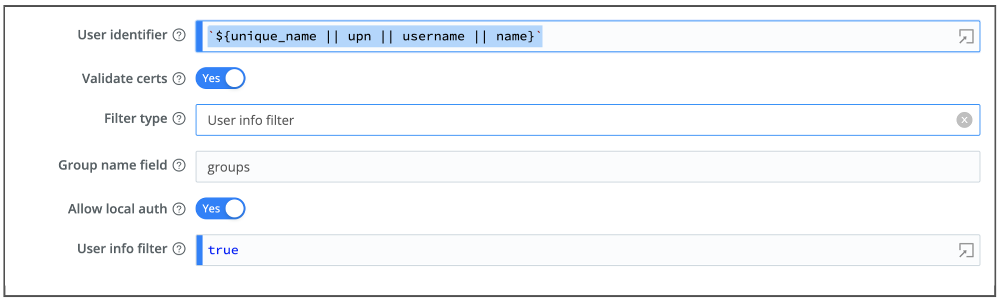
- **Type**: **OpenID Connect**. This will expose relevant fields, setting several default values and placeholders.
- **Provider name**: Enter an arbitrary identifier for this Azure AD integration.

- **Audience**: Enter your Cribl Stream Leader instance's base URL. Use the format: `https://<your-domain.ext>:9000` (do not use a trailing slash).
- **Client ID**: Enter your Microsoft Entra ID **Application (client) ID**. (In the Azure portal, see above to copy this from your app's **Overview** page.)
- **Client secret**: Enter the **Client secret** > **Value** that you earlier generated and copied from the Azure app's **Certificates & secrets** page.
- **Scope**: Accept the default `openid profile email` scopes.
- **Authentication URL**: Paste the **OAuth 2.0 authorization endpoint** that you copied above from the Azure app's **Overview** > **Endpoints** drawer.
- **Token URL**: Paste the **OAuth 2.0 token endpoint** that you copied above from the Azure app's **Overview** > **Endpoints** drawer.
- **User Info URL**, **Logout URL**: Leave both fields blank.
- **User identifier**: Adjust this based on the endpoint you choose (v1 or v2) above. In v2, the `preferred_username`, `name`, and `email` fields are set, matching this field's default values. In v1, only the `name` field is included in the token by default, so an acceptable entry here might be: `` `${unique_name || upn || username || name}` ``. You can check the token fields returned by enabling debug-level logging on Cribl Stream's `auth:sso` channel.
- Change the **Filter type** to `User info filter`.
- Optionally, enable **Allow local auth** as a fallback login method.


Authentication Settings

Type*	OpenID Connect
Provider name	Azure AD
Audience*	https://leader.cribl.io:9000
Client ID*	**
Client secret*	••
Scope	openid profile email
Authentication URL*	https://login.microsoftonline.com/************************************/oauth2/authorize
Token URL*	https://login.microsoftonline.com/************************************/oauth2/token
User Info URL	Enter user info url
Logout URL	Enter logout url
User identifier	`${preferred_username || name || email}`
Validate certs	Yes
Filter type	User info filter
Group name field	groups
Allow local auth	Yes
User info filter	true

Sample Cribl Stream Authentication Settings for Azure AD (v2 endpoints)



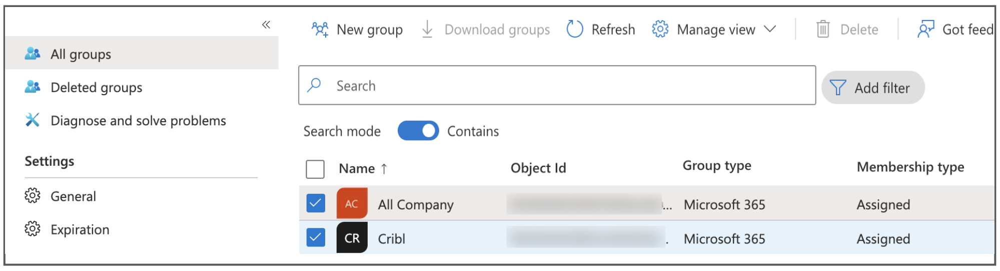
User identifier	`` `${unique_name || upn || username || name}` ``
Validate certs	Yes
Filter type	User info filter
Group name field	groups
Allow local auth	Yes
User info filter	`true`

Sample User identifier entry for v1 endpoints

Map Microsoft Entra ID Groups to Cribl Stream Roles

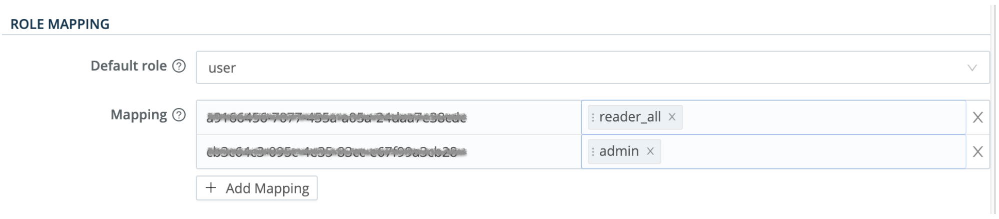
Next, map your Microsoft Entra ID groups to Cribl Stream Roles. The group names might appear as GUIDs. You can translate these on the Microsoft Entra ID **Groups** page.

Unless you synchronize Azure AD with your on-premises Active Directory, you will need to obtain the Group GUIDs from the Azure AD **Groups** page. Place these GUIDs in the mappings box and then choose the appropriate Cribl Stream Role. Here is a simple example.



Name ↑	Object Id	Group type	Membership type
AC All Company	[redacted]	Microsoft 365	Assigned
CR Cribl	[redacted]	Microsoft 365	Assigned

Microsoft Entra ID groups...



ROLE MAPPING

Default role	user	
Mapping	[redacted]	reader_all
	[redacted]	admin

...mapped to Cribl Stream Roles

17.6.2.3. Azure Event Hubs Integrations

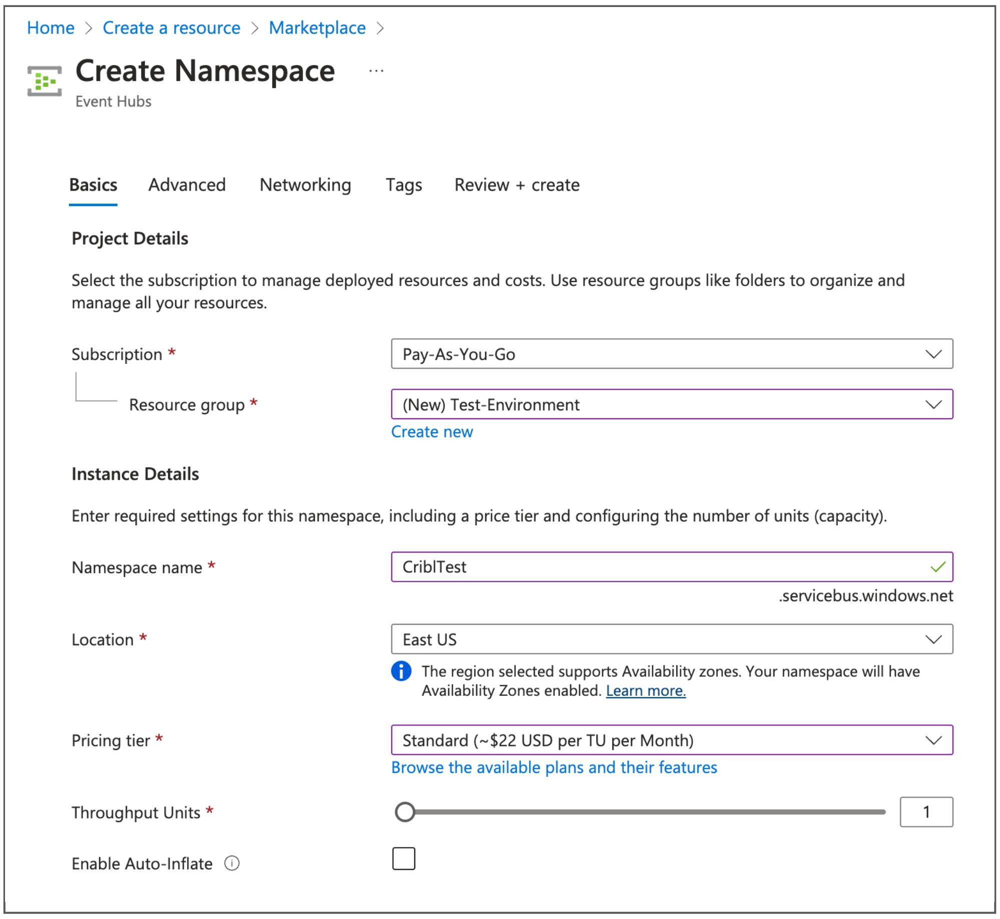
To ingest data from an Azure Event Hub, you can create an Event Hub and a Cribl Stream Azure Event Hubs Source configured to consume data from the Event Hub. This page documents that integration.

Your Azure Event Hubs account must be at the Standard (or a higher) pricing tier, because the Basic pricing tier does not support the PLAIN authentication method Cribl Stream uses for Event Hubs.

Prepare Azure Event Hubs to Send Data to Cribl Stream

First, create an Azure Event Hub as described in the Azure documentation.

In this tutorial, we use `CriblTest` as the **Namespace name**. You can use any **Namespace name** you like, as long as it is unique across Azure.



Project Details

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription *	Pay-As-You-Go
Resource group *	(New) Test-Environment

Instance Details

Enter required settings for this namespace, including a price tier and configuring the number of units (capacity).

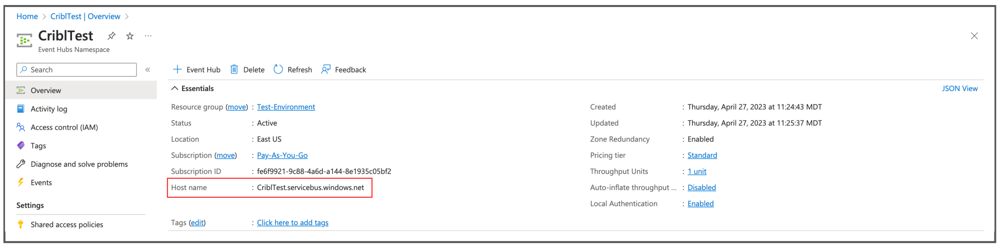
Namespace name *	CriblTest .servicebus.windows.net
Location *	East US
Pricing tier *	Standard (~$22 USD per TU per Month)
Throughput Units *	1
Enable Auto-Inflate	☐

The region selected supports Availability zones. Your namespace will have Availability Zones enabled. Learn more.

Creating an Event Hubs Namespace

Collect the information you will need when configuring Cribl Stream:

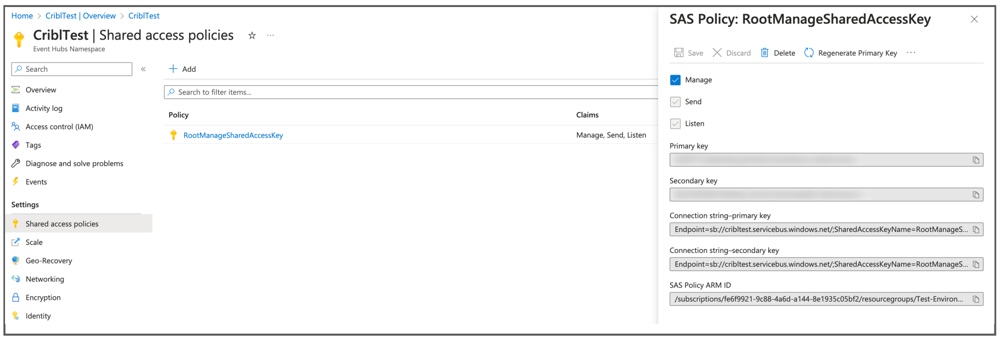
1. In the **Deployment** page, click **Go to Resource**.
2. Copy the **Host Name** and store it.



Resource group (move) : Test-Environment
Status : Active
Location : East US
Subscription (move) : Pay-As-You-Go
Subscription ID : fe6f9921-9c88-4a6d-a144-8e1935c05bf2
Host name : CriblTest.servicebus.windows.net
Tags (edit) : Click here to add tags
Created : Thursday, April 27, 2023 at 11:24:43 MDT
Updated : Thursday, April 27, 2023 at 11:25:37 MDT
Zone Redundancy : Enabled
Pricing tier : Standard
Throughput Units : 1 unit
Auto-inflate throughput ... : Disabled
Local Authentication : Enabled

Finding the Host name and Shared access policies

3. Click **Shared Access Policies** to open the page where you can select policies for your Event Hubs Namespace, and then click `RootManageSharedAccessKey` to show details for that policy.



Policy	Claims
RootManageSharedAccessKey	Manage, Send, Listen

[✓] Manage
[✓] Send
[✓] Listen

Primary key
[redacted]

Secondary key
[redacted]

Connection string–primary key
Endpoint=sb://cribltest.servicebus.windows.net/;SharedAccessKeyName=RootManageS...

Connection string–secondary key
Endpoint=sb://cribltest.servicebus.windows.net/;SharedAccessKeyName=RootManageS...

SAS Policy ARM ID
/subscriptions/fe6f9921-9c88-4a6d-a144-8e1935c05bf2/resourcegroups/Test-Environ...

Viewing Shared access policies

4. Copy and securely store the **Connection String-primary key**.
5. Click on Event Hubs on the left-hand column and copy the Event Hubs name you want to ingest.

Restricting Public Network Access for Cribl.Cloud Deployments

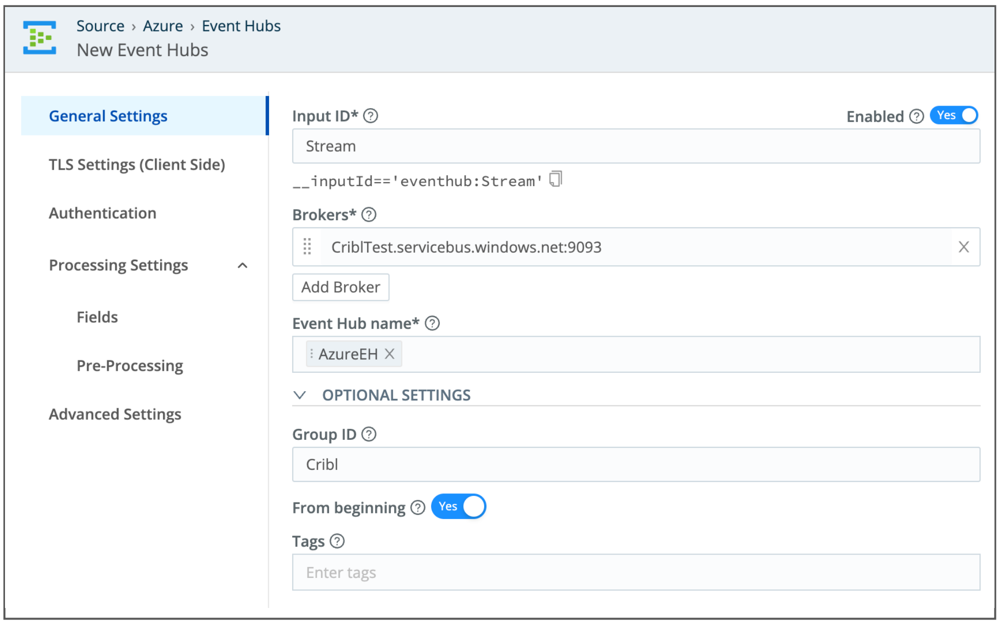
If you are using a Cribl.Cloud deployment, you have the option of restricting public network access to the Event Hubs environment.

To restrict access, you will need to obtain the egress IP in your Cribl.Cloud portal, then use it to configure public network access in the Event Hubs environment.

1. To obtain the egress IP, click **Access Details** and copy the **Egress IP** of your Cribl.Cloud Organization.
2. In the Azure Event Hubs environment, select **Networking** > **Public network access** > **Selected networks**.
3. On that same page, paste the Cribl.Cloud egress IP into the **Address range** field in the Event Hubs environment.

Configure the Azure Event Hubs Source in Cribl Stream

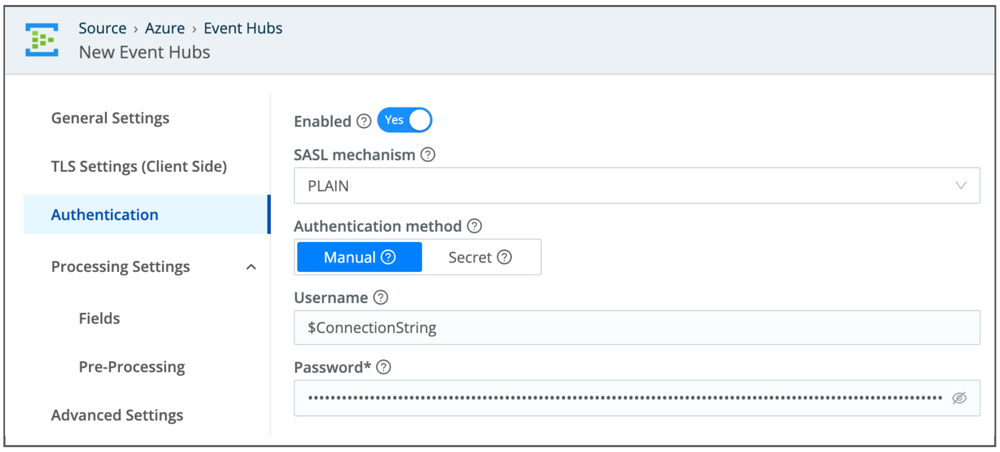
1. From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Sources**, then select **Azure Event Hubs** from the **Manage Sources** page's tiles or left nav. Click **New Source** to open the **Azure Event Hubs** > **New Source** modal.
2. In the **General Settings** tab, enter the following values:
   - **InputId**: An **InputId** of your choice. For example: `Stream`.
   - **Brokers**: End the Host Name you wrote down earlier and append port `9093`. For this example: `CriblTest.servicebus.windows.net:9093`.
   - **Event Hub Name**: The name of your Azure Event Hub. For this example: `AzureEH`. The **Event Hub Name** is equivalent to a Kafka topic. It should not be the same as the Azure Event Hubs namespace value you created above.
   - **Group ID**: This value can be `Cribl`, but the **Group ID** cannot be shared with other technologies. Sharing this value across technologies may cause unexpected behavior, such as failing to collect logs reliably.


Source › Azure › Event Hubs
New Event Hubs

Input ID*: Stream
Enabled: Yes
__inputId=='eventhub:Stream'
Brokers*: CriblTest.servicebus.windows.net:9093
Add Broker
Event Hub name*: AzureEH
OPTIONAL SETTINGS
Group ID: Cribl
From beginning: Yes
Tags: Enter tags

The General Settings tab

3. In the **Authentication** tab, enter or select the following values:
   - **SASL mechanism**: Options include `PLAIN` and `OAUTHBEARER`.
   - **Username**: `$ConnectionString` (the default generated by Azure).
   - **Authentication Method**: Select `Manual` to use the Connection String Key generated by Azure Event Hubs.
   - **Password**: Enter the **Connection String-primary key** that you copied earlier.


Source › Azure › Event Hubs
New Event Hubs

Enabled: Yes
SASL mechanism: PLAIN
Authentication method: [✓ Manual] Secret
Username: $ConnectionString
Password*: ••

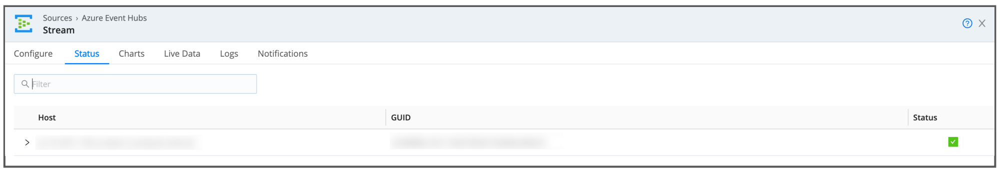
The Authentication tab

Verify that Data is Flowing from Your Azure Event Hub to Cribl Stream

Before you can verify that data is reaching Cribl Stream, you must ensure that it is flowing out of your Azure Event Hub in the first place.

One option is to configure a Datagen and a Route to send data to the Event Hub destination. We'll assume you have done that, or gotten data flowing from your Azure Event Hub in some other way.

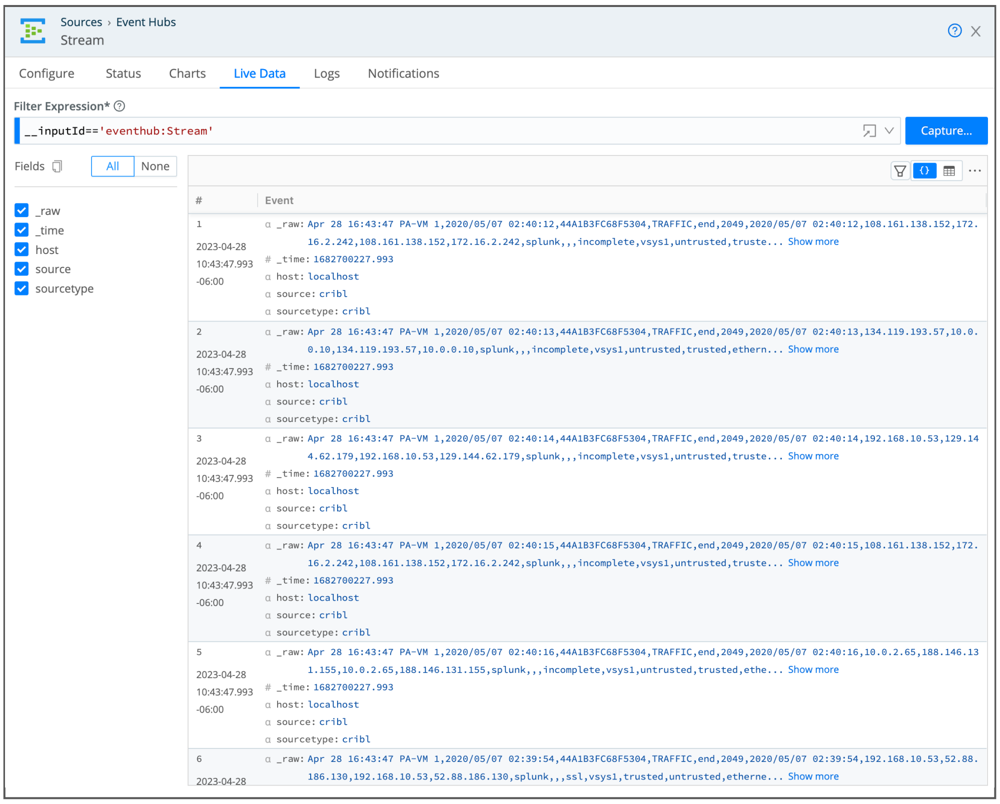
1. In Cribl Stream, open the **Sources** > **Azure Event Hubs** > **Cribl Stream** page. This should show your Source, with a message confirming that it is working properly.


Sources › Azure Event Hubs
Stream

Host	GUID	Status
[redacted]	[redacted]	✓

A working Source

2. Open the **Live Data** tab. You should see the data that is flowing from your Azure Event Hub to Cribl Stream.



```
__inputId=='eventhub:Stream'
```

- [✓] _raw
- [✓] _time
- [✓] host
- [✓] source
- [✓] sourcetype

```
#                Event
1                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:40:12,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:40:12,108.161.138.152,172.
2023-04-28               16.2.242,108.161.138.152,172.16.2.242,splunk,,,incomplete,vsys1,untrusted,truste... Show more
10:43:47.993     # _time: 1682700227.993
-06:00           α host: localhost
                 α source: cribl
                 α sourcetype: cribl
2                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:40:13,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:40:13,134.119.193.57,10.0.
2023-04-28               0.10,134.119.193.57,10.0.0.10,splunk,,,incomplete,vsys1,untrusted,trusted,ethern... Show more
10:43:47.993     # _time: 1682700227.993
-06:00           α host: localhost
                 α source: cribl
                 α sourcetype: cribl
3                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:40:14,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:40:14,192.168.10.53,129.14
2023-04-28               4.62.179,192.168.10.53,129.144.62.179,splunk,,,incomplete,vsys1,untrusted,truste... Show more
10:43:47.993     # _time: 1682700227.993
-06:00           α host: localhost
                 α source: cribl
                 α sourcetype: cribl
4                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:40:15,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:40:15,108.161.138.152,172.
2023-04-28               16.2.242,108.161.138.152,172.16.2.242,splunk,,,incomplete,vsys1,untrusted,truste... Show more
10:43:47.993     # _time: 1682700227.993
-06:00           α host: localhost
                 α source: cribl
                 α sourcetype: cribl
5                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:40:16,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:40:16,10.0.2.65,188.146.13
2023-04-28               1.155,10.0.2.65,188.146.131.155,splunk,,,incomplete,vsys1,untrusted,trusted,ethe... Show more
10:43:47.993     # _time: 1682700227.993
-06:00           α host: localhost
                 α source: cribl
                 α sourcetype: cribl
6                α _raw: Apr 28 16:43:47 PA-VM 1,2020/05/07 02:39:54,44A1B3FC68F5304,TRAFFIC,end,2049,2020/05/07 02:39:54,192.168.10.53,52.88.
2023-04-28               186.130,192.168.10.53,52.88.186.130,splunk,,,ssl,vsys1,trusted,untrusted,etherne... Show more
```

Viewing Live Data

17.6.2.4. Azure Sentinel SIEM Integration

You can send events through Cribl Stream Azure Sentinel Destinations to the Microsoft Sentinel SIEM in Azure.

These docs use the terms "Azure Sentinel" and "Microsoft Sentinel" interchangeably.

This topic explains the overall workflow for getting data from Cribl Stream into Microsoft Sentinel. While we'll cover key points about the Cribl Stream Azure Sentinel Destination here, you should refer to the Sentinel Destination topic for complete UI details.

Before You Begin

Ensure that you have prepared your Azure Workspace. You must complete this step before you can send events through the Azure Sentinel Destination to the Microsoft Sentinel SIEM.

If you do not yet have Microsoft Sentinel up and running, please join us on Cribl's Community Slack at https://cribl-community.slack.com/ and let us know; very likely, a wise goat will offer time-saving suggestions.

Event Types

Of the many event types (also called "tables") present in the Microsoft ecosystem, this topic provides what you need to get started quickly with any or all of the following four:

- CommonSecurityLog
- SecurityEvent
- Syslog
- WindowsEvent

We'll also explain how you can bring additional event types into your deployment.

To plan your Cribl/Sentinel integration, start by deciding which event types (a.k.a. tables) you want to send to Sentinel.

If the answer is "more than one," you should use an Output Router and multiple Destinations. You'll configure one Destination per table, but a single URL will ingest all the data. The Output Router can then route a given event type to the correct Destination.

The four event types above belong to the much larger set of event types that the Azure Monitor provides in its Logs Ingestion API. You can work with any of these event types as you would the four types above. See the Appendix below for the full list.

Creating Data Collection Rules

The Sentinel SIEM can have multiple ingestion endpoints, called Data Collection Endpoints (DCEs). In turn, each DCE can have multiple Data Collection Rules (DCRs). A DCR can dictate the structure of the data flowing into one table, or multiple tables.

Cribl's template is an example of a custom DCR that sends data to four different tables, one for each of the four event types enumerated above.

Alternatively, you could populate one table with different kinds of data. For example, you could define a table for the Syslog event type, and then create a **separate** DCR for Windows, Linux, and Mac system logs, respectively, all of which would populate that single table.

Besides the Azure documentation, you can study the Cribl template to understand how DCRs are structured. For example, each table's columns are defined in a sub-element of the `resources` > `properties` > `streamDeclarations` element. The `dataFlows` element – at the same level as `streamDeclarations` – specifies the table that a given type of data should populate, via the `outputStream` sub-element. It also specifies the query that will translate incoming fields into values in the table's columns, via the `transformKql` sub-element.

Read more about DCRs in the Azure Sentinel documentation.

Custom DCR Template Errors

If you get an error such as `Table for output stream 'Microsoft-CommonSecurityLog' is not available for destination 'logAnalyticsWorkspace'`), it is probably because you don't have Sentinel assigned to your Log Analytics workspace. Please add Microsoft Sentinel to your Log Analytics Workspace to correct this error.

Obtaining Your URL

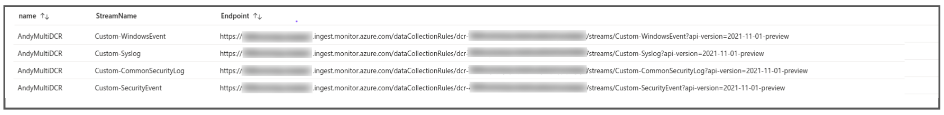
To configure your Cribl Stream Azure Sentinel Destination, you must know the ingestion URL to send data to. This is in **General Settings** > **Endpoint configuration**, where you choose between the **URL** and **ID** options. The **ID** option is just a somewhat roundabout method for generating the URL – in this section, we'll ignore **ID** in favor of an easier way to obtain the URL.

In your Azure portal:

1. Navigate to the Azure Resource Graph Explorer. The Resource Graph Explorer queries across your whole Azure cloud.
2. Run the following query in the Explorer to list all your Data Collection Endpoints (DCEs) and Data Collection Rules (DCRs).

```
Resources
| where type =~ 'microsoft.insights/datacollectionrules'
| mv-expand Streams= properties['dataFlows']
| project name, id, DCE = tostring(properties['dataCollectionEndpointId']), In
 | join kind=leftouter (Resources
| where type =~ 'microsoft.insights/datacollectionendpoints'
| project name,  DCE = tostring(id), endpoint = properties['logsIngestion']['e
| project name, StreamName, Endpoint = strcat(endpoint, '/dataCollectionRules/
```

3. If you are using all four of the tables mentioned above, the output should look something like this:



name ↑↓	StreamName	Endpoint ↑↓
AndyMultiDCR	Custom-WindowsEvent	https://[redacted].ingest.monitor.azure.com/dataCollectionRules/dcr-[redacted]/streams/Custom-WindowsEvent?api-version=2021-11-01-preview
AndyMultiDCR	Custom-Syslog	https://[redacted].ingest.monitor.azure.com/dataCollectionRules/dcr-[redacted]/streams/Custom-Syslog?api-version=2021-11-01-preview
AndyMultiDCR	Custom-CommonSecurityLog	https://[redacted].ingest.monitor.azure.com/dataCollectionRules/dcr-[redacted]/streams/Custom-CommonSecurityLog?api-version=2021-11-01-preview
AndyMultiDCR	Custom-SecurityEvent	https://[redacted].ingest.monitor.azure.com/dataCollectionRules/dcr-[redacted]/streams/Custom-SecurityEvent?api-version=2021-11-01-preview

Azure Resource Graph Explorer query output

Problems with Resource Graph Explorer Queries

If you get zero returns or the query returns unexpected data, be sure you have properly prepared your Azure workspace. This step is a preprequisite.

4. Click a DCE of interest to open its **Details** drawer.
5. Find the **Endpoint** field and click its **Copy to clipboard** icon.
   - Microsoft defines the Endpoint URI as follows:

```
{Data Collection Endpoint URI}/dataCollectionRules/{DCR Immutable ID}/streams/
```

6. In your Cribl Stream Azure Sentinel Destination, paste the copied Endpoint URI into the **URL** field.

Configuring Your Destinations, Continued

In most respects, Cribl Stream's Azure Sentinel and Webhook Destinations are alike. This section notes the few differences, while highlighting key points about configuring Azure Sentinel Destinations.

In Azure Sentinel Destination, **General Settings** > **Authentication** offers fewer options than the Webhook Destination. This is because OAuth is Cribl Stream's only supported option for Microsoft Sentinel, for now.

Some fields that present in other Destinations always have the same values for Microsoft Sentinel, and Cribl Stream fills those out for you without exposing them in the UI.

The Azure Sentinel Destination does not support client-side TLS.

The value for **Advanced Settings** > **Max body size (KB)** value defaults to `1000`. The intent here is to keep request bodies from exceeding Microsoft Sentinel's limit of 1 MB, meaning 1024 KB including headers. If necessary, you can lower **Max body size (KB)** even further.

Appendix: Azure Monitor Logs Ingestion API Event Types

Here's the whole list – note that this includes event types for AWS and for Google Cloud Platform (GCP).

Category	Event Type
Active Directory	ADAssessmentRecommendation
	ADSecurityAssessmentRecommendation
Advanced Security Information Model (ASIM)	ASimAuditEventLogs
	ASimAuthenticationEventLogs
	ASimDhcpEventLogs
	ASimDnsActivityLogs
	ASimDnsAuditLogs
	ASimFileEventLogs
	ASimNetworkSessionLogs
	ASimProcessEventLogs
	ASimRegistryEventLogs
	ASimUserManagementActivityLogs
	ASimWebSessionLogs
AWS	AWSCloudTrail
	AWSCloudWatch

Category	Event Type
	AWSGuardDuty
	AWSVPCFlow
Azure	AzureAssessmentRecommendation
Common Event Format (CEF)	CommonSecurityLog
Microsoft TVM (Defender for Endpoints)	DeviceTvmSecureConfigurationAssessmentKB
	DeviceTvmSoftwareVulnerabilitiesKB
Microsoft Exchange	ExchangeAssessmentRecommendation
	ExchangeOnlineAssessmentRecommendation
Google Cloud Platform	GCPAuditLogs
	GoogleCloudSCC
Microsoft SCCM (System Center Configuration Manager)	SCCMAssessmentRecommendation
Microsoft System Center Operations Manager (SCOM)	SCOMAssessmentRecommendation
Security events collected from windows machines by Azure Security Center or Azure Sentinel	SecurityEvent
Skype for Business Online	SfBAssessmentRecommendation
	SfBOnlineAssessmentRecommendation
SharePoint	SharePointOnlineAssessmentRecommendation
Workloads	SPAssessmentRecommendation
	SQLAssessmentRecommendation
StorageInsights	StorageInsightsAccountPropertiesDaily
	StorageInsightsDailyMetrics
	StorageInsightsHourlyMetrics
	StorageInsightsMonthlyMetrics
	StorageInsightsWeeklyMetrics

Category	Event Type
Syslog	Syslog
Update Compliance	UCClient
	UCClientReadinessStatus
	UCClientUpdateStatus
	UCDeviceAlert
	UCDOAggregatedStatus
	UCDOStatus
	UCServiceUpdateStatus
	UCUpdateAlert
Windows	WindowsClientAssessmentRecommendation
	WindowsEvent
	WindowsServerAssessmentRecommendation

17.6.2.4.1. DCR Template With Stream Values

```
{
    "$schema": "https://schema.management.azure.com/schemas/2019-04-01/deploymentTe
    "contentVersion": "1.0.0.0",
    "parameters": {
        "dataCollectionRuleName": {
            "type": "string",
            "metadata": {
                "description": "Specifies the name of the Data Collection Rule to c
            }
        },
        "location": {
            "defaultValue": "[resourceGroup().location]",
            "type": "string",
            "metadata": {
                "description": "Specifies the location in which to create the Data
            }
        },
        "workspaceResourceId": {
            "type": "string",
            "metadata": {
                "description": "Specifies the Azure resource ID of the Log Analytic
            }
        },
        "endpointResourceId": {
            "type": "string",
            "metadata": {
                "description": "Specifies the Azure resource ID of the Data Collect
            }
        }
    },
    "resources": [{
        "type": "Microsoft.Insights/dataCollectionRules",
        "apiVersion": "2021-09-01-preview",
        "name": "[parameters('dataCollectionRuleName')]",
        "location": "[parameters('location')]",
        "properties": {
            "dataCollectionEndpointId": "[parameters('endpointResourceId')]",
            "streamDeclarations": {
                "Custom-CommonSecurityLog": {
                    "columns": [{
                            "name": "Activity",
                            "type": "string"
```

```
},
{
    "name": "AdditionalExtensions",
    "type": "string"
},
{
    "name": "ApplicationProtocol",
    "type": "string"
},
{
    "name": "CommunicationDirection",
    "type": "string"
},
{
    "name": "Computer",
    "type": "string"
},
{
    "name": "DestinationDnsDomain",
    "type": "string"
},
{
    "name": "DestinationHostName",
    "type": "string"
},
{
    "name": "DestinationIP",
    "type": "string"
},
{
    "name": "DestinationMACAddress",
    "type": "string"
},
{
    "name": "DestinationNTDomain",
    "type": "string"
},
{
    "name": "DestinationPort",
    "type": "int"
},
{
    "name": "DestinationProcessId",
    "type": "int"
},
{
    "name": "DestinationProcessName",
    "type": "string"
},
{
    "name": "DestinationServiceName",
    "type": "string"
},
{
    "name": "DestinationTranslatedAddress",
    "type": "string"
},
{
    "name": "DestinationTranslatedPort",
    "type": "int"
},
{
    "name": "DestinationUserID",
    "type": "string"
},
{
    "name": "DestinationUserName",
    "type": "string"
},
{
    "name": "DestinationUserPrivileges",
    "type": "string"
},
{
    "name": "DeviceAction",
    "type": "string"
},
{
    "name": "DeviceAddress",
    "type": "string"
},
{
    "name": "DeviceCustomDate1",
    "type": "string"
},
{
    "name": "DeviceCustomDate1Label",
    "type": "string"
},
{
    "name": "DeviceCustomDate2",
    "type": "string"
},
{
    "name": "DeviceCustomDate2Label",
    "type": "string"
},
{
    "name": "DeviceCustomFloatingPoint1",
    "type": "real"
},
{
    "name": "DeviceCustomFloatingPoint1Label",
    "type": "string"
},
{
    "name": "DeviceCustomFloatingPoint2",
    "type": "real"
},
{
    "name": "DeviceCustomFloatingPoint2Label",
    "type": "string"
},
{
    "name": "DeviceCustomFloatingPoint3",
    "type": "real"
},
{
    "name": "DeviceCustomFloatingPoint3Label",
    "type": "string"
},
{
    "name": "DeviceCustomFloatingPoint4",
    "type": "real"
},
{
    "name": "DeviceCustomFloatingPoint4Label",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address1",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address1Label",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address2",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address2Label",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address3",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address3Label",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address4",
    "type": "string"
},
{
    "name": "DeviceCustomIPv6Address4Label",
    "type": "string"
},
{
    "name": "DeviceCustomNumber1",
    "type": "int"
},
{
    "name": "DeviceCustomNumber1Label",
    "type": "string"
},
{
    "name": "DeviceCustomNumber2",
    "type": "int"
```

```
},
{
    "name": "DeviceCustomNumber2Label",
    "type": "string"
},
{
    "name": "DeviceCustomNumber3",
    "type": "int"
},
{
    "name": "DeviceCustomNumber3Label",
    "type": "string"
},
{
    "name": "DeviceCustomString1",
    "type": "string"
},
{
    "name": "DeviceCustomString1Label",
    "type": "string"
},
{
    "name": "DeviceCustomString2",
    "type": "string"
},
{
    "name": "DeviceCustomString2Label",
    "type": "string"
},
{
    "name": "DeviceCustomString3",
    "type": "string"
},
{
    "name": "DeviceCustomString3Label",
    "type": "string"
},
{
    "name": "DeviceCustomString4",
    "type": "string"
},
{
    "name": "DeviceCustomString4Label",
    "type": "string"
},
{
    "name": "DeviceCustomString5",
    "type": "string"
},
{
    "name": "DeviceCustomString5Label",
    "type": "string"
},
{
    "name": "DeviceCustomString6",
    "type": "string"
},
{
    "name": "DeviceCustomString6Label",
    "type": "string"
},
{
    "name": "DeviceDnsDomain",
    "type": "string"
},
{
    "name": "DeviceEventCategory",
    "type": "string"
},
{
    "name": "DeviceEventClassID",
    "type": "string"
},
{
    "name": "DeviceExternalID",
    "type": "string"
},
{
    "name": "DeviceFacility",
    "type": "string"
},
{
    "name": "DeviceInboundInterface",
    "type": "string"
},
{
    "name": "DeviceMacAddress",
    "type": "string"
},
{
    "name": "DeviceName",
    "type": "string"
},
{
    "name": "DeviceNtDomain",
    "type": "string"
},
{
    "name": "DeviceOutboundInterface",
    "type": "string"
},
{
    "name": "DevicePayloadId",
    "type": "string"
},
{
    "name": "DeviceProduct",
    "type": "string"
},
{
    "name": "DeviceTimeZone",
    "type": "string"
},
{
    "name": "DeviceTranslatedAddress",
    "type": "string"
},
{
    "name": "DeviceVendor",
    "type": "string"
},
{
    "name": "DeviceVersion",
    "type": "string"
},
{
    "name": "EndTime",
    "type": "datetime"
},
{
"name": "EventCount",
"type": "int"
},
{
"name": "EventOutcome",
"type": "string"
},
{
"name": "EventType",
"type": "int"
},
{
"name": "ExternalID",
"type": "int"
},
{
"name": "ExtID",
"type": "string"
},
{
"name": "FieldDeviceCustomNumber1",
"type": "long"
},
{
"name": "FieldDeviceCustomNumber2",
"type": "long"
},
{
"name": "FieldDeviceCustomNumber3",
"type": "long"
},
{
"name": "FileCreateTime",
"type": "string"
},
{
"name": "FileHash",
"type": "string"
},
{
"name": "FileID",
"type": "string"
```

```
},
{
    "name": "FileModificationTime",
    "type": "string"
},
{
    "name": "FileName",
    "type": "string"
},
{
    "name": "FilePath",
    "type": "string"
},
{
    "name": "FilePermission",
    "type": "string"
},
{
    "name": "FileSize",
    "type": "int"
},
{
    "name": "FileType",
    "type": "string"
},
{
    "name": "FlexDate1",
    "type": "string"
},
{
    "name": "FlexDate1Label",
    "type": "string"
},
{
    "name": "FlexNumber1",
    "type": "int"
},
{
    "name": "FlexNumber1Label",
    "type": "string"
},
{
    "name": "FlexNumber2",
    "type": "int"
},
{
    "name": "FlexNumber2Label",
    "type": "string"
},
{
    "name": "FlexString1",
    "type": "string"
},
{
    "name": "FlexString1Label",
    "type": "string"
},
{
    "name": "FlexString2",
    "type": "string"
},
{
    "name": "FlexString2Label",
    "type": "string"
},
{
    "name": "IndicatorThreatType",
    "type": "string"
},
{
    "name": "LogSeverity",
    "type": "string"
},
{
    "name": "MaliciousIP",
    "type": "string"
},
{
    "name": "MaliciousIPCountry",
    "type": "string"
},
{
    "name": "MaliciousIPLatitude",
    "type": "real"
},
{
    "name": "MaliciousIPLongitude",
    "type": "real"
},
{
    "name": "Message",
    "type": "string"
},
{
    "name": "OldFileCreateTime",
    "type": "string"
},
{
    "name": "OldFileHash",
    "type": "string"
},
{
    "name": "OldFileID",
    "type": "string"
},
{
    "name": "OldFileModificationTime",
    "type": "string"
},
{
    "name": "OldFileName",
    "type": "string"
},
{
    "name": "OldFilePath",
    "type": "string"
},
{
    "name": "OldFilePermission",
    "type": "string"
},
{
    "name": "OldFileSize",
    "type": "int"
},
{
    "name": "OldFileType",
    "type": "string"
},
{
    "name": "OriginalLogSeverity",
    "type": "string"
},
{
    "name": "ProcessID",
    "type": "int"
},
{
    "name": "ProcessName",
    "type": "string"
},
{
    "name": "Protocol",
    "type": "string"
},
{
    "name": "Reason",
    "type": "string"
},
{
    "name": "ReceiptTime",
    "type": "string"
},
{
    "name": "ReceivedBytes",
    "type": "long"
},
{
    "name": "RemoteIP",
    "type": "string"
},
{
    "name": "RemotePort",
    "type": "string"
},
{
    "name": "ReportReferenceLink",
    "type": "string"
},
{
    "name": "RequestClientApplication",
    "type": "string"
```

```
        },
        {
            "name": "RequestContext",
            "type": "string"
        },
        {
            "name": "RequestCookies",
            "type": "string"
        },
        {
            "name": "RequestMethod",
            "type": "string"
        },
        {
            "name": "RequestURL",
            "type": "string"
        },
        {
            "name": "SentBytes",
            "type": "long"
        },
        {
            "name": "SimplifiedDeviceAction",
            "type": "string"
        },
        {
            "name": "SourceDnsDomain",
            "type": "string"
        },
        {
            "name": "SourceHostName",
            "type": "string"
        },
        {
            "name": "SourceIP",
            "type": "string"
        },
        {
            "name": "SourceMACAddress",
            "type": "string"
        },
        {
            "name": "SourceNTDomain".
            "type": "string"
        },
        {
            "name": "SourcePort",
            "type": "int"
        },
        {
            "name": "SourceProcessId",
            "type": "int"
        },
        {
            "name": "SourceProcessName",
            "type": "string"
        },
        {
            "name": "SourceServiceName",
            "type": "string"
        },
        {
            "name": "SourceSystem",
            "type": "string"
        },
        {
            "name": "SourceTranslatedAddress",
            "type": "string"
        },
        {
            "name": "SourceTranslatedPort",
            "type": "int"
        },
        {
            "name": "SourceUserID",
            "type": "string"
        },
        {
            "name": "SourceUserName",
            "type": "string"
        },
        {
            "name": "SourceUserPrivileges",
            "type": "string"
        },
        {
            "name": "StartTime",
            "type": "datetime"
        },
        {
            "name": "ThreatConfidence",
            "type": "string"
        },
        {
            "name": "ThreatDescription",
            "type": "string"
        },
        {
            "name": "ThreatSeverity",
            "type": "int"
        },
        {
            "name": "TimeGenerated",
            "type": "datetime"
        }
    ]
    },
    "Custom-SecurityEvent": {
        "columns": [{
            "name": "AccessList",
            "type": "string"
        },
        {
            "name": "AccessMask",
            "type": "string"
        },
        {
            "name": "AccessReason",
            "type": "string"
        },
        {
            "name": "Account",
            "type": "string"
        },
        {
            "name": "AccountDomain",
            "type": "string"
        },
        {
```

```
{
    "name": "AccountExpires",
    "type": "string"
},
{
    "name": "AccountName",
    "type": "string"
},
{
    "name": "AccountSessionIdentifier",
    "type": "string"
},
{
    "name": "AccountType",
    "type": "string"
},
{
    "name": "Activity",
    "type": "string"
},
{
    "name": "AdditionalInfo",
    "type": "string"
},
{
    "name": "AdditionalInfo2",
    "type": "string"
},
{
    "name": "AllowedToDelegateTo",
    "type": "string"
},
{
    "name": "Attributes",
    "type": "string"
},
{
    "name": "AuditPolicyChanges",
    "type": "string"
},
{
    "name": "AuditsDiscarded",
    "type": "int"
}
```

```
},
{
    "name": "AuthenticationLevel",
    "type": "int"
},
{
    "name": "AuthenticationPackageName",
    "type": "string"
},
{
    "name": "AuthenticationProvider",
    "type": "string"
},
{
    "name": "AuthenticationServer",
    "type": "string"
},
{
    "name": "AuthenticationService",
    "type": "int"
},
{
    "name": "AuthenticationType",
    "type": "string"
},
{
    "name": "AzureDeploymentID",
    "type": "string"
},
{
    "name": "CACertificateHash",
    "type": "string"
},
{
    "name": "CallerProcessId",
    "type": "string"
},
{
    "name": "CalledStationID",
    "type": "string"
},
{
    "name": "CallerProcessName",
    "type": "string"
    "type": "string"
},
{
    "name": "CallingStationID",
    "type": "string"
},
{
    "name": "CAPublicKeyHash",
    "type": "string"
},
{
    "name": "CategoryId",
    "type": "string"
},
{
    "name": "CertificateDatabaseHash",
    "type": "string"
},
{
    "name": "Channel",
    "type": "string"
},
{
    "name": "ClassId",
    "type": "string"
},
{
    "name": "ClassName",
    "type": "string"
},
{
    "name": "ClientAddress",
    "type": "string"
},
{
    "name": "ClientIPAddress",
    "type": "string"
},
{
    "name": "ClientName",
    "type": "string"
},
{
    "name": "CommandLine"
```

```
        "name": "CommandLine",
        "type": "string"
    },
    {
        "name": "CompatibleIds",
        "type": "string"
    },
    {
        "name": "Computer",
        "type": "string"
    },
    {
        "name": "DCDNSName",
        "type": "string"
    },
    {
        "name": "DeviceId",
        "type": "string"
    },
    {
        "name": "DisplayName",
        "type": "string"
    },
    {
        "name": "Disposition",
        "type": "string"
    },
    {
        "name": "DomainBehaviorVersion",
        "type": "string"
    },
    {
        "name": "DomainName",
        "type": "string"
    },
    {
        "name": "DomainPolicyChanged",
        "type": "string"
    },
    {
        "name": "DomainSid",
        "type": "string"
    },
{
    "name": "EAPType",
    "type": "string"
},
{
    "name": "ErrorCode",
    "type": "int"
},
{
    "name": "ElevatedToken",
    "type": "string"
},
{
    "name": "EventID",
    "type": "int"
},
{
    "name": "EventData",
    "type": "string"
},
{
    "name": "EventSourceName",
    "type": "string"
},
{
    "name": "ExtendedQuarantineState",
    "type": "string"
},
{
    "name": "FailureReason",
    "type": "string"
},
{
    "name": "FileHash",
    "type": "string"
},
{
    "name": "FilePath",
    "type": "string"
},
{
    "name": "FilePathNoUser",
    "type": "string"
}
```

```
},
{
    "name": "Filter",
    "type": "string"
},
{
    "name": "ForceLogoff",
    "type": "string"
},
{
    "name": "Fqbn",
    "type": "string"
},
{
    "name": "FullyQualifiedSubjectMachineName",
    "type": "string"
},
{
    "name": "FullyQualifiedSubjectUserName",
    "type": "string"
},
{
    "name": "GroupMembership",
    "type": "string"
},
{
    "name": "HandleId",
    "type": "string"
},
{
    "name": "HardwareIds",
    "type": "string"
},
{
    "name": "HomeDirectory",
    "type": "string"
},
{
    "name": "HomePath",
    "type": "string"
},
{
    "name": "ImpersonationLevel",
    "type": "string"
},
{
    "name": "IpAddress",
    "type": "string"
},
{
    "name": "IpPort",
    "type": "string"
},
{
    "name": "KeyLength",
    "type": "int"
},
{
    "name": "Level",
    "type": "string"
},
{
    "name": "LmPackageName",
    "type": "string"
},
{
    "name": "LocationInformation",
    "type": "string"
},
{
    "name": "LockoutDuration",
    "type": "string"
},
{
    "name": "LockoutObservationWindow",
    "type": "string"
},
{
    "name": "LockoutThreshold",
    "type": "string"
},
{
    "name": "LoggingResult",
    "type": "string"
},
{
    "name": "LogonHours",
    "type": "string"
},
{
    "name": "LogonID",
    "type": "string"
},
{
    "name": "LogonProcessName",
    "type": "string"
},
{
    "name": "LogonType",
    "type": "int"
},
{
    "name": "LogonTypeName",
    "type": "string"
},
{
    "name": "MachineAccountQuota",
    "type": "string"
},
{
    "name": "MachineInventory",
    "type": "string"
},
{
    "name": "MachineLogon",
    "type": "string"
},
{
    "name": "ManagementGroupName",
    "type": "string"
},
{
    "name": "MandatoryLabel",
    "type": "string"
},
{
    "name": "MaxPasswordAge",
    "type": "string"
```

```
},
{
    "name": "MemberName",
    "type": "string"
},
{
    "name": "MemberSid",
    "type": "string"
},
{
    "name": "MinPasswordAge",
    "type": "string"
},
{
    "name": "MinPasswordLength",
    "type": "string"
},
{
    "name": "MixedDomainMode",
    "type": "string"
},
{
    "name": "NASIdentifier",
    "type": "string"
},
{
    "name": "NASIPv4Address",
    "type": "string"
},
{
    "name": "NASIPv6Address",
    "type": "string"
},
{
    "name": "NASPort",
    "type": "string"
},
{
    "name": "NASPortType",
    "type": "string"
},
{
    "name": "NetworkPolicyName",
        "type": "string"
    },
    {
        "name": "NewDate",
        "type": "string"
    },
    {
        "name": "NewMaxUsers",
        "type": "string"
    },
    {
        "name": "NewProcessId",
        "type": "string"
    },
    {
        "name": "NewProcessName",
        "type": "string"
    },
    {
        "name": "NewRemark",
        "type": "string"
    },
    {
        "name": "NewShareFlags",
        "type": "string"
    },
    {
        "name": "NewTime",
        "type": "string"
    },
    {
        "name": "NewUacValue",
        "type": "string"
    },
    {
        "name": "NewValue",
        "type": "string"
    },
    {
        "name": "NewValueType",
        "type": "string"
    },
    {
        "name": "ObjectName",
        "type": "string"
    },
    {
        "name": "ObjectServer",
        "type": "string"
    },
    {
        "name": "ObjectType",
        "type": "string"
    },
    {
        "name": "ObjectValueName",
        "type": "string"
    },
    {
        "name": "OemInformation",
        "type": "string"
    },
    {
        "name": "OldMaxUsers",
        "type": "string"
    },
    {
        "name": "OldRemark",
        "type": "string"
    },
    {
        "name": "OldShareFlags",
        "type": "string"
    },
    {
        "name": "OldUacValue",
        "type": "string"
    },
    {
        "name": "OldValue",
        "type": "string"
    },
    {
        "name": "OldValueType",
        "type": "string"
    },
{
    "name": "OperationType",
    "type": "string"
},
{
    "name": "PackageName",
    "type": "string"
},
{
    "name": "ParentProcessName",
    "type": "string"
},
{
    "name": "PartitionKey",
    "type": "string"
},
{
    "name": "PasswordHistoryLength",
    "type": "string"
},
{
    "name": "PasswordLastSet",
    "type": "string"
},
{
    "name": "PasswordProperties",
    "type": "string"
},
{
    "name": "PreviousDate",
    "type": "string"
},
{
    "name": "PreviousTime",
    "type": "string"
},
{
    "name": "PrimaryGroupId",
    "type": "string"
},
{
    "name": "PrivateKeyUsageCount",
    "type": "string"
```

```
},
{
    "name": "PrivilegeList",
    "type": "string"
},
{
    "name": "Process",
    "type": "string"
},
{
    "name": "ProcessId",
    "type": "string"
},
{
    "name": "ProcessName",
    "type": "string"
},
{
    "name": "ProfilePath",
    "type": "string"
},
{
    "name": "Properties",
    "type": "string"
},
{
    "name": "ProtocolSequence",
    "type": "string"
},
{
    "name": "ProxyPolicyName",
    "type": "string"
},
{
    "name": "QuarantineHelpURL",
    "type": "string"
},
{
    "name": "QuarantineSessionID",
    "type": "string"
},
{
    "name": "QuarantineSessionIdentifier",
    "type": "string"
},
{
    "name": "QuarantineState",
    "type": "string"
},
{
    "name": "QuarantineSystemHealthResult",
    "type": "string"
},
{
    "name": "RelativeTargetName",
    "type": "string"
},
{
    "name": "RemoteIpAddress",
    "type": "string"
},
{
    "name": "RemotePort",
    "type": "string"
},
{
    "name": "Requester",
    "type": "string"
},
{
    "name": "RequestId",
    "type": "string"
},
{
    "name": "RestrictedAdminMode",
    "type": "string"
},
{
    "name": "RowKey",
    "type": "string"
},
{
    "name": "RowsDeleted",
    "type": "string"
},
{
    "name": "SamAccountName",
    "type": "string"
},
{
    "name": "ScriptPath",
    "type": "string"
},
{
    "name": "SecurityDescriptor",
    "type": "string"
},
{
    "name": "ServiceAccount",
    "type": "string"
},
{
    "name": "ServiceFileName",
    "type": "string"
},
{
    "name": "ServiceName",
    "type": "string"
},
{
    "name": "ServiceStartType",
    "type": "int"
},
{
    "name": "ServiceType",
    "type": "string"
},
{
    "name": "SessionName",
    "type": "string"
},
{
    "name": "ShareLocalPath",
    "type": "string"
},
{
    "name": "ShareName",
    "type": "string"
},
{
    "name": "SidHistory",
    "type": "string"
},

{
    "name": "SourceSystem",
    "type": "string"
},
{
    "name": "Status",
    "type": "string"
},
{
    "name": "StorageAccount",
    "type": "string"
},
{
    "name": "SubcategoryId",
    "type": "string"
},

{
    "name": "Subject",
    "type": "string"
},
{
    "name": "SubjectAccount",
    "type": "string"
},
{
    "name": "SubjectDomainName",
    "type": "string"
},
{
    "name": "SubjectKeyIdentifier",
    "type": "string"
},
{
    "name": "SubjectLogonId",
    "type": "string"
},
{
    "name": "SubjectMachineName",
    "type": "string"
},
{
    "name": "SubjectMachineSID",
    "type": "string"
},
{
    "name": "SubjectUserName",
    "type": "string"
},
{
    "name": "SubjectUserSid",
    "type": "string"
},
{
    "name": "SubStatus",
    "type": "string"
},
{
    "name": "TableId",
    "type": "string"
},
{
    "name": "TargetDomainName",
    "type": "string"
},
{
    "name": "TargetInfo",
    "type": "string"
},
{
    "name": "TargetAccount",
    "type": "string"
},
{
    "name": "TargetLinkedLogonId",
    "type": "string"
},
{
    "name": "TargetLogonId",
    "type": "string"
},
{
    "name": "TargetOutboundDomainName",
    "type": "string"
},
{
    "name": "TargetOutboundUserName",
    "type": "string"
},
{
    "name": "TargetServerName",
    "type": "string"
},
{
    "name": "TargetSid",
    "type": "string"
},
{
    "name": "TargetUser",
    "type": "string"
},
{
    "name": "TargetUserName",
    "type": "string"
},
{
    "name": "TargetUserSid",
    "type": "string"
},
{
    "name": "Task",
    "type": "int"
},
{
    "name": "TemplateContent",
    "type": "string"
},
{
    "name": "TemplateDSObjectFQDN",
    "type": "string"
},
{
    "name": "TemplateInternalName",
    "type": "string"
},
{
    "name": "TemplateOID",
    "type": "string"
},
{
    "name": "TemplateSchemaVersion",
    "type": "string"
},
{
    "name": "TemplateVersion",
    "type": "string"
},
{
    "name": "TimeCollected",
    "type": "datetime"
},
{
    "name": "TimeGenerated",
    "type": "datetime"
},
{
    "name": "TokenElevationType",
    "type": "string"
},
{
    "name": "TransmittedServices",
    "type": "string"
},
{
    "name": "UserAccountControl",
    "type": "string"
},
{
    "name": "UserParametersUserParameters",
    "type": "string"
},
{
    "name": "UserPrincipalName",
    "type": "string"
},
{
    "name": "UserWorkstationsUserWorkstations",
            "type": "string"
        },
        {
            "name": "VendorIds",
            "type": "string"
        },
        {
            "name": "VirtualAccount",
            "type": "string"
        },
        {
            "name": "Workstation",
            "type": "string"
        },
        {
            "name": "WorkstationName",
            "type": "string"
        }
    ]
},
"Custom-Syslog": {
    "columns": [{
            "name": "Computer",
            "type": "string"
        },
        {
            "name": "EventTime",
            "type": "datetime"
        },
        {
            "name": "Facility",
            "type": "string"
        },
        {
            "name": "HostIP",
            "type": "string"
        },
        {
            "name": "HostName",
            "type": "string"
        },
        {
            "name": "ManagementGroupName",
            "type": "string"
        },
        {
            "name": "ProcessID",
            "type": "int"
        },
        {
            "name": "ProcessName",
            "type": "string"
        },
        {
            "name": "SeverityLevel",
            "type": "string"
        },
        {
            "name": "SourceSystem",
            "type": "string"
        },
        {
            "name": "SyslogMessage",
            "type": "string"
        },
        {
            "name": "TimeCollected",
            "type": "datetime"
        },
        {
            "name": "TimeGenerated",
            "type": "datetime"
        }
    ]
},
"Custom-WindowsEvent": {
    "columns": [{
            "name": "Channel",
            "type": "string"
        },
        {
            "name": "Computer",
            "type": "string"
        },
        {
            "name": "EventData",
            "type": "string"
        },
        {
            "name": "EventID",
            "type": "int"
        },
        {
            "name": "EventLevel",
            "type": "int"
        },
        {
            "name": "EventLevelName",
            "type": "string"
        },
        {
            "name": "EventOriginId",
            "type": "string"
        },
        {
            "name": "ManagementGroupName",
            "type": "string"
        },
        {
            "name": "Provider",
            "type": "string"
        },
        {
            "name": "RawEventData",
            "type": "string"
        },
        {
            "name": "SourceSystem",
            "type": "string"
        },
        {
            "name": "Task",
            "type": "int"
        },
        {
            "name": "TimeGenerated",
            "type": "datetime"
        }
    ]
```

```
        ]
    }
},
"destinations": {
    "logAnalytics": [{
        "workspaceResourceId": "[parameters('workspaceResourceId')]",
        "name": "logAnalyticsWorkspace"
    }]
},
"dataFlows": [{
        "streams": [
            "Custom-CommonSecurityLog"
        ],
        "destinations": [
            "logAnalyticsWorkspace"
        ],
        "transformKql": "source",
        "outputStream": "Microsoft-CommonSecurityLog"
    },
    {
        "streams": [
            "Custom-SecurityEvent"
        ],
        "destinations": [
            "logAnalyticsWorkspace"
        ],
        "transformKql": "source",
        "outputStream": "Microsoft-SecurityEvent"
    },
    {
        "streams": [
            "Custom-Syslog"
        ],
        "destinations": [
            "logAnalyticsWorkspace"
        ],
        "transformKql": "source",
        "outputStream": "Microsoft-Syslog"
    },
    {
        "streams": [
            "Custom-WindowsEvent"
        ],
        "destinations": [
                        "destinations": [
                            "logAnalyticsWorkspace"
                        ],
                        "transformKql": "source",
                        "outputStream": "Microsoft-WindowsEvent"
                    }
                ]
            }
        }],
        "outputs": {
            "dataCollectionRuleId": {
                "type": "string",
                "value": "[resourceId('Microsoft.Insights/dataCollectionRules', parame
            }
        }
    }
```

17.6.3. Splunk

17.6.3.1. Splunk To Elasticsearch

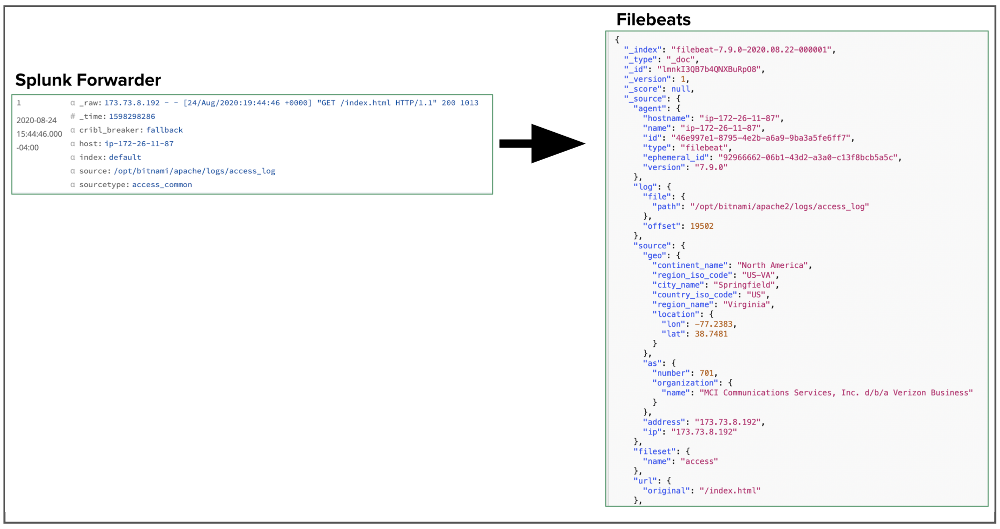
To route data from existing Splunk infrastructure to Elasticsearch services, you might face a daunting task: re-architecting your entire forwarding tier. This could require retooling lots of servers – up to hundreds, or thousands – to uninstall their Splunk forwarders, and swap in Elastic-compatible agents.

Cribl Stream can reduce this effort to just a few hours: Configure one Splunk `outputs.conf` stanza to output to Cribl Stream, and propagate that across all your Splunk servers. Done!

Next, you can easily configure Cribl Stream to listen for Splunk data on one port, and to route that data to all the Elasticsearch destinations you want to feed.

Transforming Data from Splunk to Elastic Format

Also, in Cribl Stream's core, you can easily design a Pipeline that modifies the original Splunk event into Elastic's Common Schema – making it look exactly like an event generated by an Elastic agent.
These transformations help you make the most of Elastic's offerings, like Filebeats, etc.



```
1                 α _raw: 173.73.8.192 - - [24/Aug/2020:19:44:46 +0000] "GET /index.html HTTP/1.1" 200 1013
2020-08-24        # _time: 1598298286
15:44:46.000      α cribl_breaker: fallback
-04:00            α host: ip-172-26-11-87
                  α index: default
                  α source: /opt/bitnami/apache/logs/access_log
                  α sourcetype: access_common
```

```
{
  "_index": "filebeat-7.9.0-2020.08.22-000001",
  "_type": "_doc",
  "_id": "lmnkI3QB7b4QNXBuRpO8",
  "_version": 1,
  "_score": null,
  "_source": {
    "agent": {
      "hostname": "ip-172-26-11-87",
      "name": "ip-172-26-11-87",
      "id": "46e997e1-8795-4e2b-a6a9-9ba3a5fe6ff7",
      "type": "filebeat",
      "ephemeral_id": "92966662-06b1-43d2-a3a0-c13f8bcb5a5c",
      "version": "7.9.0"
    },
    "log": {
      "file": {
        "path": "/opt/bitnami/apache2/logs/access_log"
      },
      "offset": 19502
    },
    "source": {
      "geo": {
        "continent_name": "North America",
        "region_iso_code": "US-VA",
        "city_name": "Springfield",
        "country_iso_code": "US",
        "region_name": "Virginia",
        "location": {
          "lon": -77.2383,
          "lat": 38.7481
        }
      },
      "as": {
        "number": 701,
        "organization": {
          "name": "MCI Communications Services, Inc. d/b/a Verizon Business"
        }
      },
      "address": "173.73.8.192",
      "ip": "173.73.8.192"
    },
    "fileset": {
      "name": "access"
    },
    "url": {
      "original": "/index.html"
    },
```

Transforming to Elastic Common Schema

Some of the Cribl Stream Functions useful in transforming Splunk-generated events into Elastic's format are:

- Regex Extract: Extract a portion of the raw event, and place it into a specified field.
- Lookup: key off the host IP to add fields like `hostname`, `name`, `id`, and `type`.
- Eval: Turn key-value pairs into nested JSON objects.

- GeoIP: Correlate source IP to a geographic database.

We'll show all four in our example Pipeline below, although you might need only a subset.

Goat Rid of Some Guesswork

Cribl Stream will offer you further time savings as you configure the internal data transformation. Cribl Stream's Data Preview features enable you to test transformations' results as you build your Pipeline, before you commit or run it.

This eliminates blind guesswork in Splunk configuration files to specify source -> index transformations, check the results, and then start all over again. In particular, Cribl Stream's Regex Extract Function provides a regex101-like UI, to facilitate precisely designing and debugging your regex expressions.

Let's goat started on the example.

Configure Splunk Forwarder

First, in a Splunk App, configure a Splunk forwarder (UF or HF) to specify your Cribl Workers as destinations. Use outputs.conf stanzas of this form:

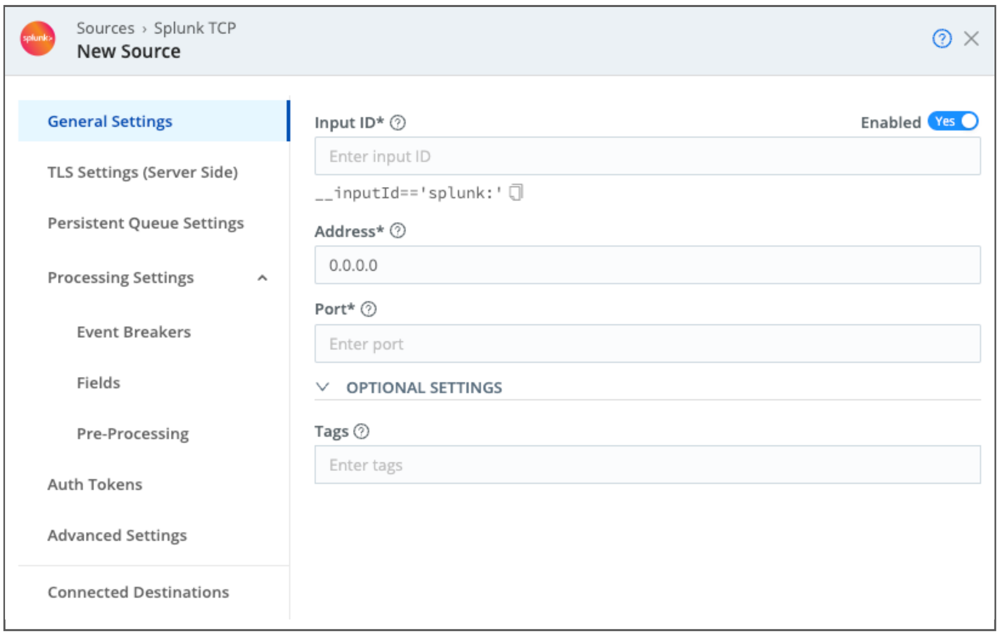
.../outputs.conf

```
[tcpout]
disabled = false
defaultGroup = cribl, <optional_clone_target_group>, ...

[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ...
sendCookedData=true
```

Push the app using the deployment server.

Configure Splunk Source in Cribl Stream

Next, in Cribl Stream, configure a Splunk Source. The key requirement here is to set the **Port** to listen on. (Optionally, you can also configure TLS, Event Breakers, metadata fields, and/or a pre-processing Pipeline.)


Sources › Splunk TCP
New Source

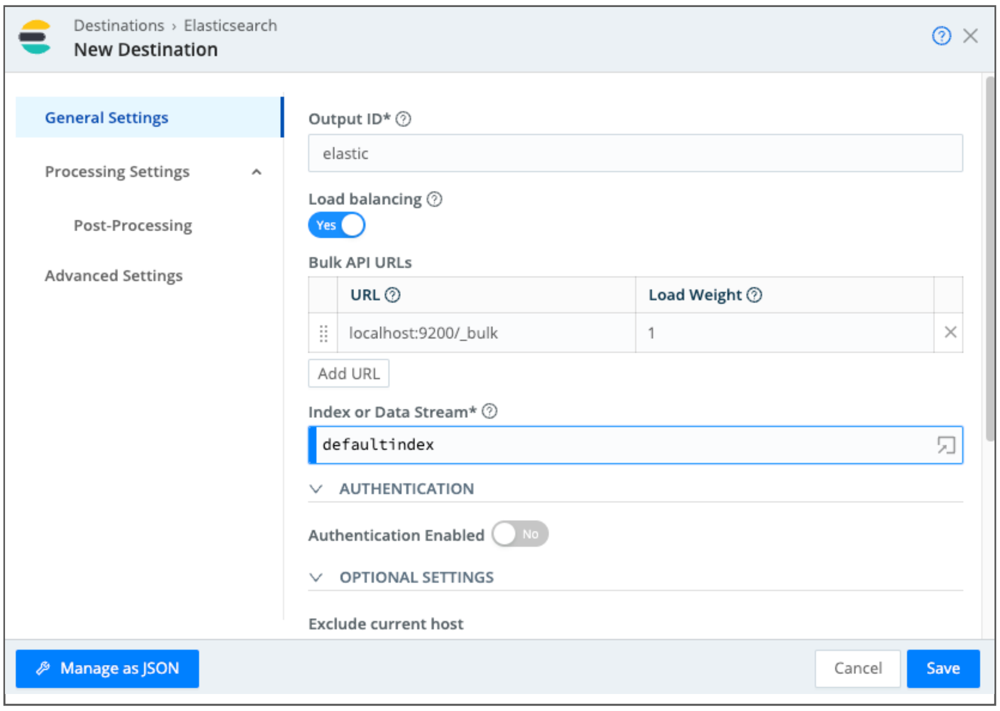
General Settings
TLS Settings (Server Side)
Persistent Queue Settings
Processing Settings
Event Breakers
Fields
Pre-Processing
Auth Tokens
Advanced Settings
Connected Destinations

Input ID* — Enter input ID — Enabled: Yes
`__inputId=='splunk:'`
Address*: 0.0.0.0
Port*: Enter port
OPTIONAL SETTINGS
Tags: Enter tags

Splunk Source configuration

Configure Elasticsearch Destination

To configure Cribl Stream’s output, set up an Elasticsearch Destination by specifying the **Bulk API URL** and **Index**.


Destinations › Elasticsearch
New Destination

General Settings
Processing Settings
Post-Processing
Advanced Settings

Output ID*: elastic
Load balancing: Yes
Bulk API URLs

URL	Load Weight
localhost:9200/_bulk	1

Add URL
Index or Data Stream*: `defaultindex`
AUTHENTICATION
Authentication Enabled: No
OPTIONAL SETTINGS
Exclude current host

Manage as JSON — Cancel — Save

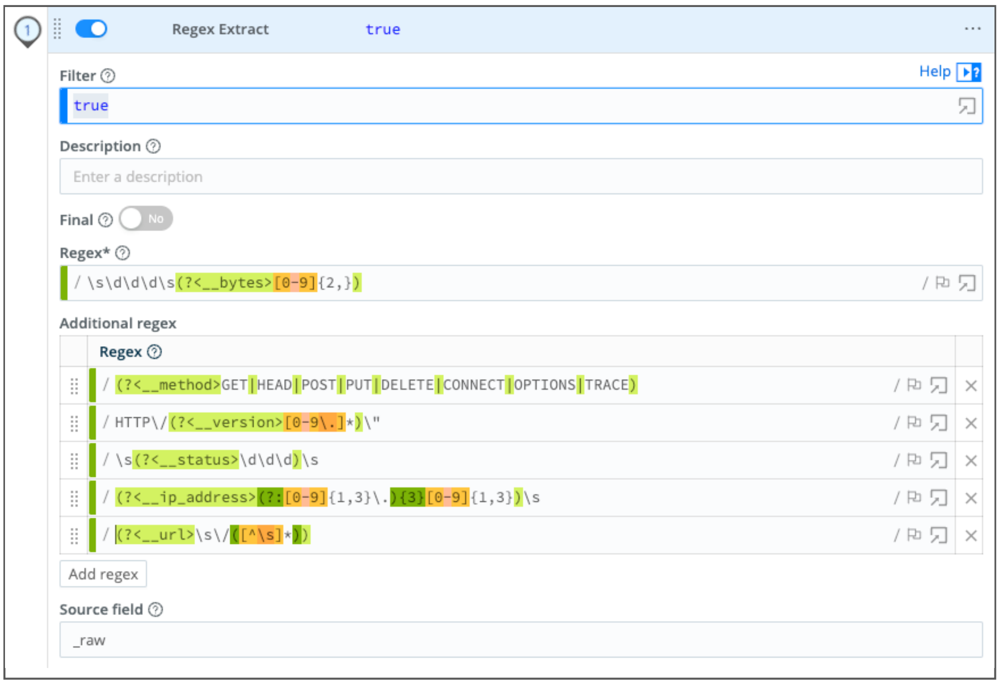
Elasticsearch Destination configuration

Configure Pipeline

Next, this section shows several Functions that you can assemble into a Pipeline to transform incoming Splunk events to match the Elastic Common Schema.

Regex Extract Function

First, use a Regex Extract Function to break the Splunk events into fields. Try the sample configuration shown below:


Regex Extract
true
Filter
Help
true
Description
Enter a description
Final
No
Regex*
/ \s\d\d\d\s(?<__bytes>[0-9]{2,})
Additional regex
Regex
/ (?<__method>GET|HEAD|POST|PUT|DELETE|CONNECT|OPTIONS|TRACE)
/ HTTP\/(?<__version>[0-9\.]*)\"
/ \s(?<__status>\d\d\d)\s
/ (?<__ip_address>(?:[0-9]{1,3}\.){3}[0-9]{1,3})\s
/ (?<__url>\s\/([^\s]*))
Add regex
Source field
_raw

Regex Extract Function

Here are the six rows of regex in this example:

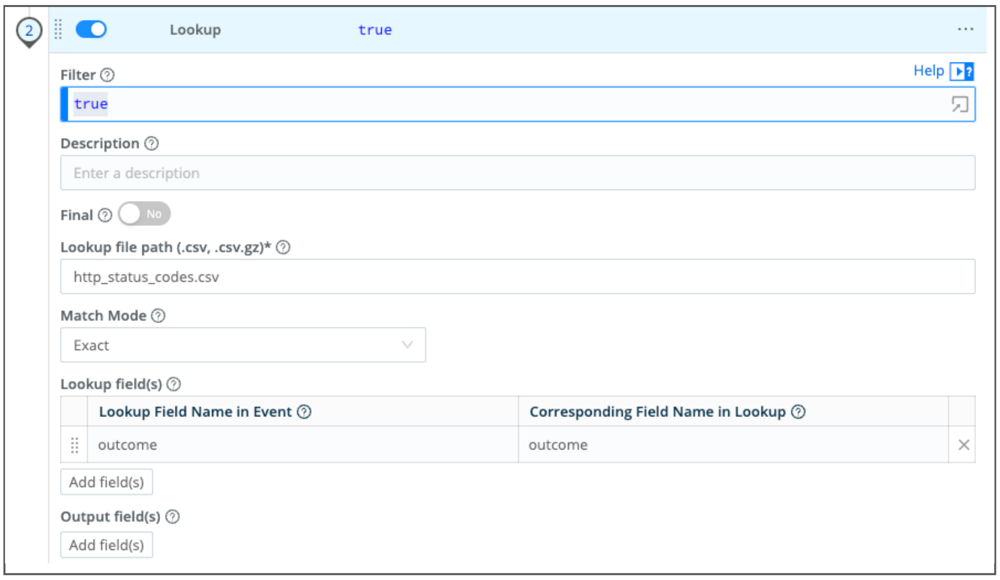
Regex; Additional Regex

```
/\s\d\d\d\s(?<__bytes>[0-9]{2,})/
/(?<__method>GET|HEAD|POST|PUT|DELETE|CONNECT|OPTIONS|TRACE)/
/HTTP\/(?<__version>[0-9\.]*)\"/
/\s(?<__status>\d\d\d)\s/
/(?<__ip_address>(?:[0-9]{1,3}\.){3}[0-9]{1,3})\s/
/(?<__url>\s\/([^\s]*))/
```

As you refine your expression, capture a sample of incoming Splunk data to test your regex's results in Cribl Stream's right Preview pane.

Lookup Function

In this example, we next add a Lookup Function, to translate HTTP error codes to readable text. Note this Function's optional **Reload Period** field, in which you can define a reload interval for a lookup file whose contents refresh frequently.



Filter: true

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: http_status_codes.csv

Match Mode: Exact

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
outcome	outcome

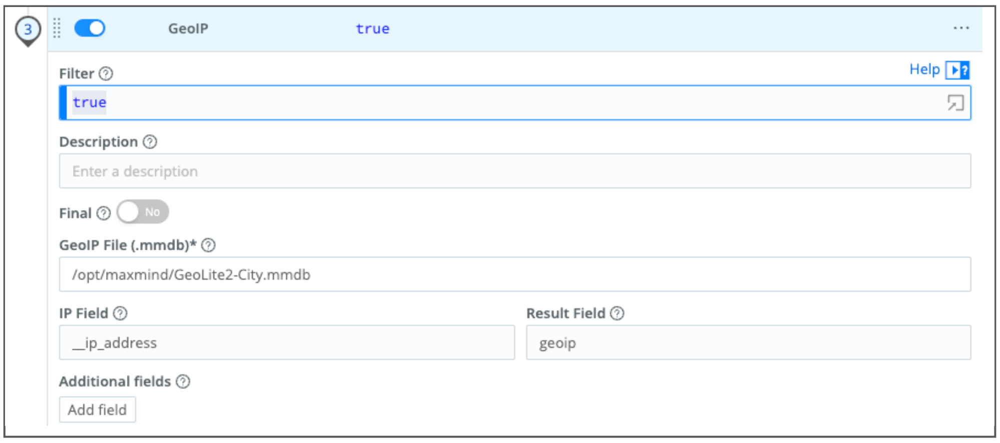
Add field(s)

Output field(s)

Add field(s)

GeoIP Function

To enrich the Splunk data, we next use a GeoIP Function. This a specialized lookup against a database of IP addresses by geographic location. This Function's output can provide Elasticsearch with `location` fields like `lat` and `long`.



Filter: true

Description: Enter a description

Final: No

GeoIP File (.mmdb)*: /opt/maxmind/GeoLite2-City.mmdb

IP Field: __ip_address

Result Field: geoip

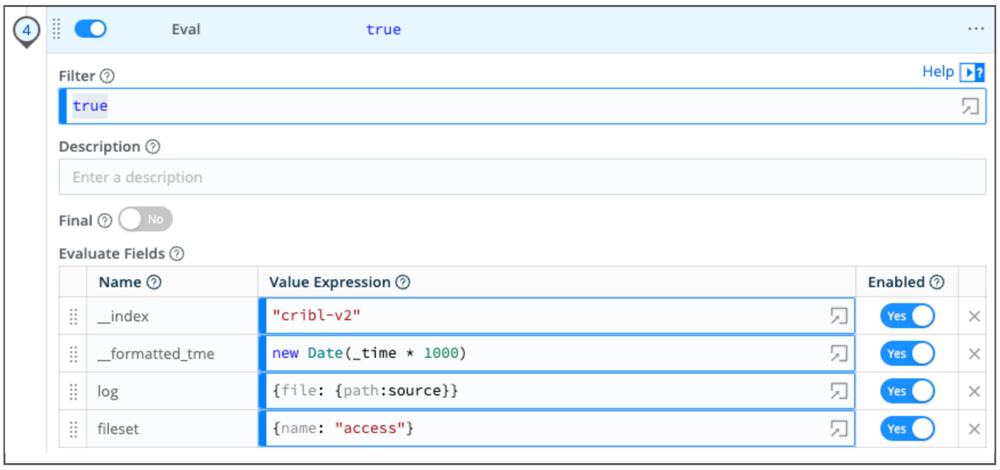
Additional fields

Add field

GeoIP specialized lookup

Eval Function

Finally, to further enrich the outbound events, the Pipeline uses an Eval Function. This adds multiple key-value pairs that define and populate fields conforming to the Elastic Common Schema.



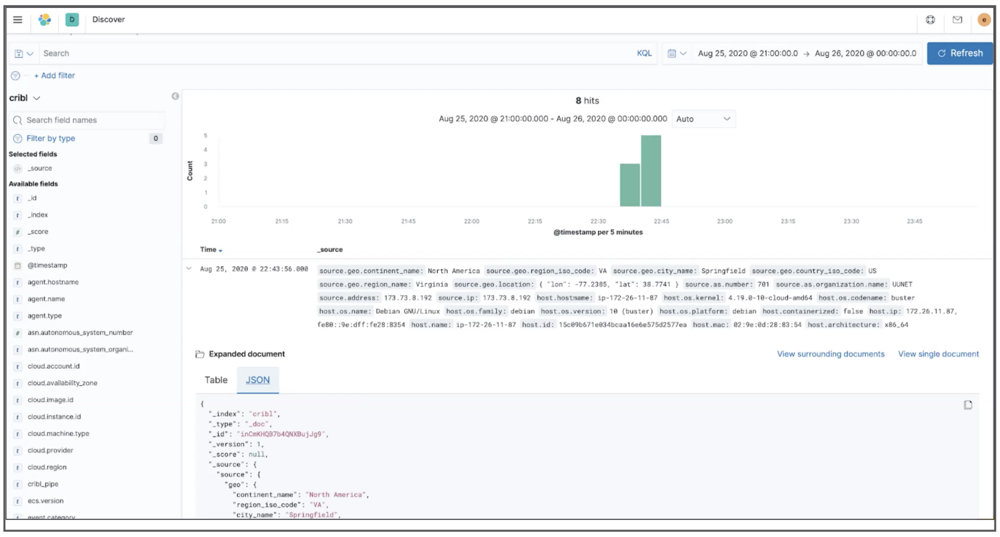
Filter: `true`

Name	Value Expression	Enabled
_index	`"cribl-v2"`	Yes
_formatted_tme	`new Date(_time * 1000)`	Yes
log	`{file: {path:source}}`	Yes
fileset	`{name: "access"}`	Yes

Eval Function

Results

After attaching your Pipeline to a Route, here's an an exported event, all happy in Elasticsearch with nested JSON.



```
Aug 25, 2020 @ 22:43:56.000  source.geo.continent_name: North America source.geo.region_iso_code: VA source.geo.city_name: Springfield source.geo.country_iso_code: US
source.geo.region_name: Virginia source.geo.location: { "lon": -77.2385, "lat": 38.7741 } source.as.number: 701 source.as.organization.name: UUNET
source.address: 173.73.8.192 source.ip: 173.73.8.192 host.hostname: ip-172-26-11-87 host.os.kernel: 4.19.0-10-cloud-amd64 host.os.codename: buster
host.os.name: Debian GNU/Linux host.os.family: debian host.os.version: 10 (buster) host.os.platform: debian host.containerized: false host.ip: 172.26.11.87,
fe80::9e:dff:fe28:8354 host.name: ip-172-26-11-87 host.id: 15c09b671e034bcaa16e6e575d2577ea host.mac: 02:9e:0d:28:83:54 host.architecture: x86_64
```

```
{
  "_index": "cribl",
  "_type": "_doc",
  "_id": "inCmKHQB7b4QNXBujJg9",
  "_version": 1,
  "_score": null,
  "_source": {
    "source": {
      "geo": {
        "continent_name": "North America",
        "region_iso_code": "VA",
        "city_name": "Springfield",
```

Event as exported to Elasticsearch

For More Info

For additional details on configuring Splunk forwarders for Cribl Stream, see this related documentation:

- Configuring a Splunk (TCP) Forwarder
- Configuring Cribl App for Splunk on an HF

17.6.3.2. Splunk To Exabeam

In many organizations, the IT department uses a tool like Splunk for operational logging, while the Security team relies on Exabeam to prevent insider threats. These tools use separate agents to access the same data, leading to some data-sharing conundrums:

- Installing the Exabeam agent in parallel with Splunk would duplicate the data.
- Some servers, like domain controllers, allow only a single agent. In this case, you can't feed two platforms with the same data.
- Querying Splunk for the data would introduce extra latency and overhead costs.
- Forwarding data directly from Splunk Universal Forwarders (UFs) is a nonstarter. Classic logs from Splunk UFs embed newlines and special characters, which break Exabeam's parser.

Cribl Stream can help you unblock these issues: Ingest data directly into Cribl Stream from Splunk UFs running on the domain controller, and transform the events in Stream before routing them to Exabeam.

While this example is written around a Splunk-to-Exabeam scenario, you can use the same general techniques to connect and transform data between several other upstream and downstream services.

Transforming Data from Splunk to Exabeam's Format

In Cribl Stream's core, you can easily design a Pipeline that modifies the original Splunk event to fit the format that Exabeam expects. Some Cribl Stream Functions useful for this transformation are:

- Serialize: Remove all of the newlines and spaces, and then transform the data into `JSON` format.
- Mask: Remove the special characters, and replace them with space.
- Eval: Create a new field called `Message`, and remove everything else using the **Remove Fields** option.

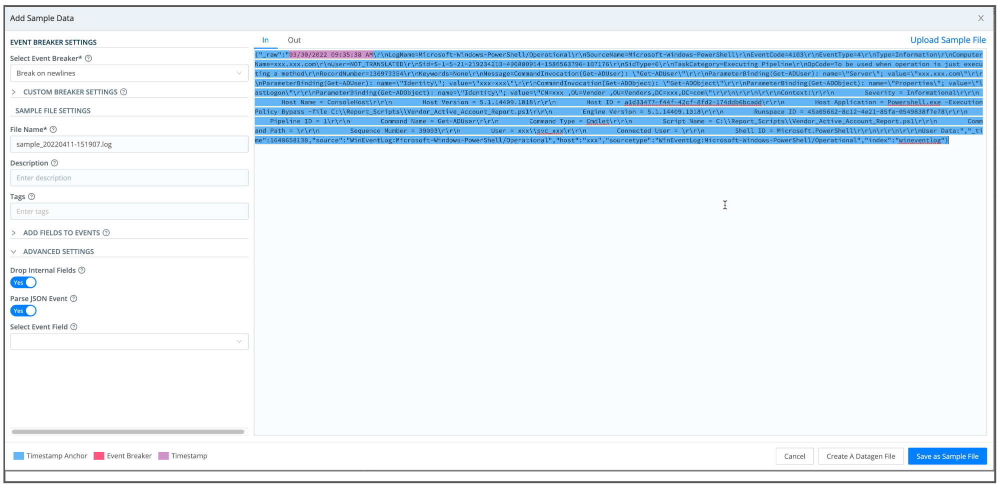
In this guide, we'll show you how to:

1. Capture sample logs.
2. Apply the Functions (outlined above) to transform the sample log into Exabeam's expected format.
3. Validate your fields with Exabeam Parsers.
4. Stream your event to Exabeam to test the entire sequence.

Capture Sample Logs

Start with sample logs of the event data you plan to work with. In our example, we'll copy and paste the log sample straight into Cribl Stream.

Once you've pasted the log sample, enter a unique **File Name** on the modal's left side, then click **Save as Sample File** at right.



Select Event Breaker*: Break on newlines

File Name*: sample_20220411-151907.log

Description: Enter description

Tags: Enter tags

Drop Internal Fields: Yes

Parse JSON Event: Yes

Select Event Field:

```
{"_raw":"03/30/2022 09:35:38 AM\r\nLogName=Microsoft-Windows-PowerShell/Operational\r\nSourceName=Microsoft-Windows-PowerShell\r\nEventCode=4103\r\nEventType=4\r\nType=Information\r\nComputer
Name=xxx.xxx.com\r\nUser=NOT_TRANSLATED\r\nSid=S-1-5-21-219234213-490800914-1586563796-107176\r\nSidType=0\r\nTaskCategory=Executing Pipeline\r\nOpCode=To be used when operation is just execu
ting a method\r\nRecordNumber=136973354\r\nKeywords=None\r\nMessage=CommandInvocation(Get-ADUser): \"Get-ADUser\"\r\r\nParameterBinding(Get-ADUser): name=\"Server\"; value=\"xxx.xxx.com\"\r\r
\nParameterBinding(Get-ADUser): name=\"Identity\"; value=\"xxx-xxx\"\r\r\nCommandInvocation(Get-ADObject): \"Get-ADObject\"\r\r\nParameterBinding(Get-ADObject): name=\"Properties\"; value=\"l
astLogon\"\r\r\nParameterBinding(Get-ADObject): name=\"Identity\"; value=\"CN=xxx ,OU=Vendor ,OU=Vendors,DC=xxx,DC=com\"\r\r\n\r\r\n\r\r\nContext:\r\r\n        Severity = Informational\r\r\n
        Host Name = ConsoleHost\r\r\n        Host Version = 5.1.14409.1018\r\r\n        Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd\r\r\n        Host Application = Powershell.exe -Execution
Policy Bypass -file C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1\r\r\n        Engine Version = 5.1.14409.1018\r\r\n        Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78\r\r\n
    Pipeline ID = 1\r\r\n        Command Name = Get-ADUser\r\r\n        Command Type = Cmdlet\r\r\n        Script Name = C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1\r\r\n        Comm
and Path = \r\r\n        Sequence Number = 39093\r\r\n        User = xxx\\svc_xxx\r\r\n        Connected User = \r\r\n        Shell ID = Microsoft.PowerShell\r\r\n\r\r\n\r\r\nUser Data:","_ti
me":1648658138,"source":"WinEventLog:Microsoft-Windows-PowerShell/Operational","host":"xxx","sourcetype":"WinEventLog:Microsoft-Windows-PowerShell/Operational","index":"wineventlog"}
```

🟦 Timestamp Anchor 🟥 Event Breaker 🟪 Timestamp

Saving captured data

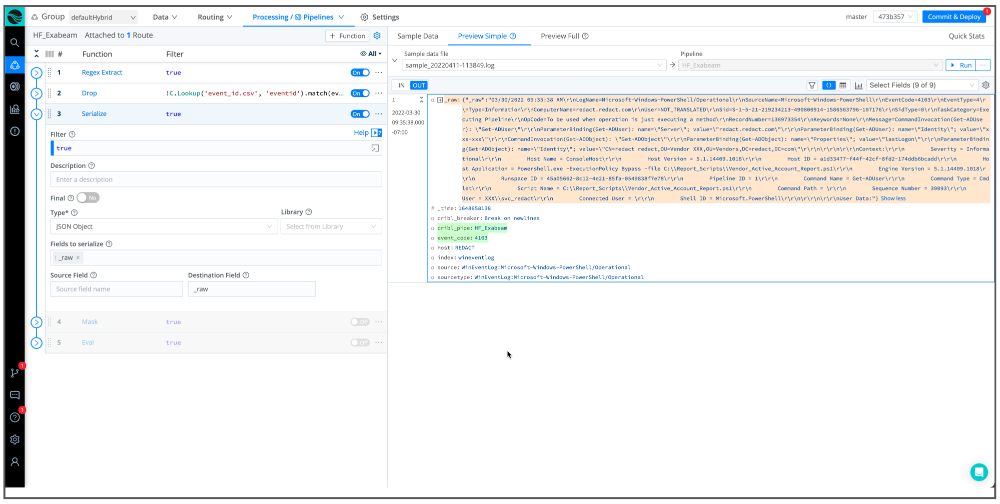
In your production environment, you can filter incoming events in real time (e.g., using a filter expression like `Windows Security logging`) to identify your in-scope Exabeam data.

Configure Pipeline

Next, this section shows relevant Functions that you can assemble into a processing Pipeline to transform the sample log into Exabeam's expected format.

Serialize Function

First, use a Serialize Function to change the event's format into `JSON`. Then, remove the extra lines and spaces, because Exabeam treats each newline as a separate event.



HF_Exabeam Attached to 1 Route

#	Function	Filter	
1	Regex Extract	`true`	On
2	Drop	`!C.Lookup('event_id.csv', 'eventid').match(ev…`	On
3	Serialize	`true`	On
4	Mask	`true`	Off
5	Eval	`true`	Off

Filter: `true`
Description: Enter a description
Final: No
Type*: JSON Object
Library: Select from Library
Fields to serialize: _raw
Source Field: Source field name
Destination Field: _raw

Sample data file: sample_20220411-113849.log → Pipeline: HF_Exabeam

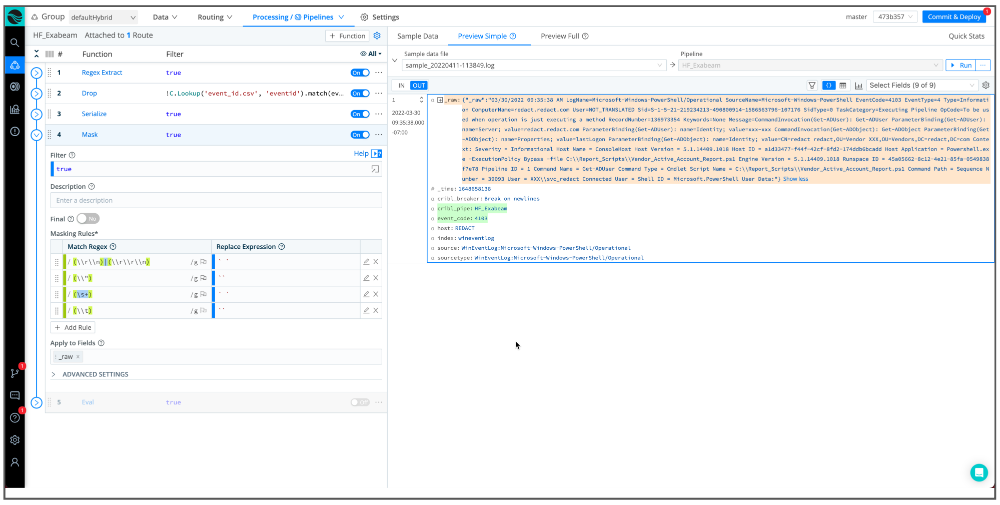
```
2022-03-30
09:35:38.000
-07:00
_raw: {"_raw":"03/30/2022 09:35:38 AM\r\nLogName=Microsoft-Windows-PowerShell/Operational\r\nSourceName=Microsoft-Windows-PowerShell\r\nEventCode=4103\r\nEventType=4\r\nType=Information\r\nComputerName=redact.redact.com\r\nUser=NOT_TRANSLATED\r\nSid=S-1-5-21-219234213-490800914-1586563796-107176\r\nSidType=0\r\nTaskCategory=Executing Pipeline\r\nOpCode=To be used when operation is just executing a method\r\nRecordNumber=136973354\r\nKeywords=None\r\nMessage=CommandInvocation(Get-ADUser): \"Get-ADUser\"\r\r\nParameterBinding(Get-ADUser): name=\"Server\"; value=\"redact.redact.com\"\r\r\nParameterBinding(Get-ADUser): name=\"Identity\"; value=\"xxx-xxx\"\r\r\nCommandInvocation(Get-ADObject): \"Get-ADObject\"\r\r\nParameterBinding(Get-ADObject): name=\"Properties\"; value=\"lastLogon\"\r\r\nParameterBinding(Get-ADObject): name=\"Identity\"; value=\"CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com\"\r\r\n\r\r\n\r\r\nContext:\r\r\n        Severity = Informational\r\r\n        Host Name = ConsoleHost\r\r\n        Host Version = 5.1.14409.1018\r\r\n        Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd\r\r\n        Host Application = Powershell.exe -ExecutionPolicy Bypass -file C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1\r\r\n        Engine Version = 5.1.14409.1018\r\r\n        Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78\r\r\n        Pipeline ID = 1\r\r\n        Command Name = Get-ADUser\r\r\n        Command Type = Cmdlet\r\r\n        Script Name = C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1\r\r\n        Command Path = \r\r\n        Sequence Number = 39093\r\r\n        User = XXX\\svc_redact\r\r\n        Connected User = \r\r\n        Shell ID = Microsoft.PowerShell\r\r\n\r\r\n\r\r\nUser Data:"} Show less
# _time: 1648658138
α cribl_breaker: Break on newlines
α cribl_pipe: HF_Exabeam
α event_code: 4103
α host: REDACT
α index: wineventlog
α source: WinEventLog:Microsoft-Windows-PowerShell/Operational
α sourcetype: WinEventLog:Microsoft-Windows-PowerShell/Operational
```

Reformatting the event

Mask Function

Next, use a Mask Function to remove extra `\n` and `\r` characters. (Otherwise, Exabeam's regex filter would reject the characters and drop whole fields that need to be matched and extracted.)

In this example, we are using a Mask Function to remove these special characters and replace them with spaces.



HF_Exabeam Attached to 1 Route

#	Function	Filter	
1	Regex Extract	`true`	On
2	Drop	`!C.Lookup('event_id.csv', 'eventid').match(ev…`	On
3	Serialize	`true`	On
4	Mask	`true`	On
5	Eval	`true`	Off

Filter: `true`
Description: Enter a description
Final: No
Masking Rules*

Match Regex	Replace Expression
`/(\\r\\n)|(\\r\\r\\n)/g`	`` ` ` ``
`/(\\")/g`	` `` `
`/(\s+)/g`	`` ` ` ``
`/(\\t)/g`	` `` `

Apply to Fields: _raw
ADVANCED SETTINGS

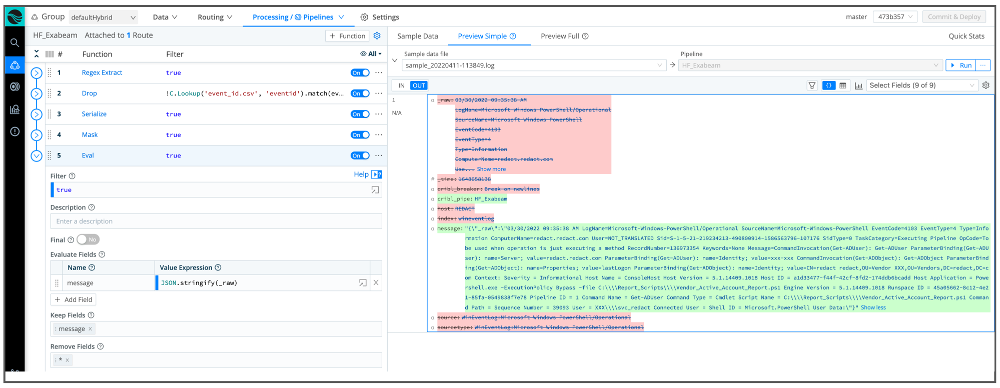
Sample data file: sample_20220411-113849.log → Pipeline: HF_Exabeam

```
2022-03-30
09:35:38.000
-07:00
_raw: {"_raw":"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=To be used when operation is just executing a method RecordNumber=136973354 Keywords=None Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding(Get-ADUser): name=Server; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject): Get-ADObject ParameterBinding(Get-ADObject): name=Properties; value=lastLogon ParameterBinding(Get-ADObject): name=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com Context: Severity = Informational Host Name = ConsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application = Powershell.exe -ExecutionPolicy Bypass -file C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet Script Name = C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1 Command Path = Sequence Number = 39093 User = XXX\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Data:"} Show less
# _time: 1648658138
α cribl_breaker: Break on newlines
α cribl_pipe: HF_Exabeam
α event_code: 4103
α host: REDACT
α index: wineventlog
α source: WinEventLog:Microsoft-Windows-PowerShell/Operational
α sourcetype: WinEventLog:Microsoft-Windows-PowerShell/Operational
```

Extraneous newlines/returns to remove

Eval Function

As the last step in Cribl Stream, use an Eval Function to enrich the outbound events, by adding key-value pair that defines and populates a field conforming to the Exabeam schema. Name the new field `Message`. (The `Message` field will populate `Raw Message` in the Exabeam Data Lake.) Remove everything else using **Remove Fields**.



#	Function	Filter	
1	Regex Extract	true	On
2	Drop	!C.Lookup('event_id.csv', 'eventid').match(ev...	On
3	Serialize	true	On
4	Mask	true	On
5	Eval	true	On

Filter: true

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
message	JSON.stringify(_raw)

Keep Fields: message

Remove Fields: *

Sample data file: sample_20220411-113849.log → Pipeline: HF_Exabeam · Run

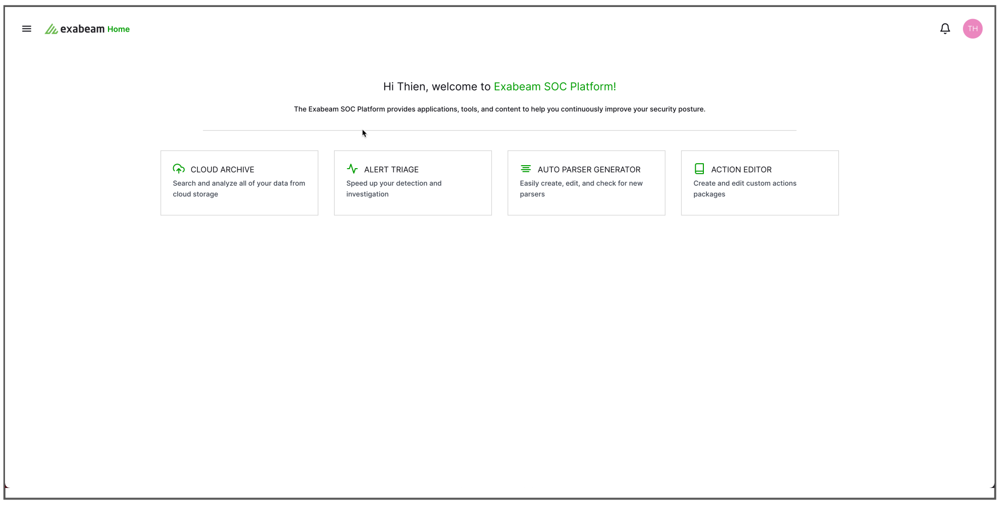
IN OUT · Select Fields (9 of 9)

```
1    ~~_raw: 03/30/2022 09:35:38 AM~~
N/A      ~~LogName=Microsoft-Windows-PowerShell/Operational~~
         ~~SourceName=Microsoft-Windows-PowerShell~~
         ~~EventCode=4103~~
         ~~EventType=4~~
         ~~Type=Information~~
         ~~ComputerName=redact.redact.com~~
         ~~Use...~~ Show more
     ~~_time: 1648658138~~
     ~~cribl_breaker: Break on newlines~~
     cribl_pipe: HF_Exabeam
     ~~host: REDACT~~
     ~~index: wineventlog~~
     message: "{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Info
              rmation ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=To
              be used when operation is just executing a method RecordNumber=136973354 Keywords=None Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding(Get-ADU
              ser): name=Server; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject): Get-ADObject ParameterBind
              ing(Get-ADObject): name=Properties; value=lastLogon ParameterBinding(Get-ADObject): name=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=c
              om Context: Severity = Informational Host Name = ConsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application = Powe
              rshell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e2
              1-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet Script Name = C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Comman
              d Path = Sequence Number = 39093 User = XXX\\\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Data:\"}" Show less
     ~~source: WinEventLog:Microsoft-Windows-PowerShell/Operational~~
     ~~sourcetype: WinEventLog:Microsoft-Windows-PowerShell/Operational~~
```

Eval Function to add an Exabeam-specific field

Goat the Fields?

Use Exabeam's Auto Parser Generator (APG) tool to validate your fields. If you don't have the APG tool, contact Exabeam support to request access via your ECP account. This tool validates matching parsers based on your event.



Hi Thien, welcome to Exabeam SOC Platform!

The Exabeam SOC Platform provides applications, tools, and content to help you continuously improve your security posture.

CLOUD ARCHIVE
Search and analyze all of your data from cloud storage

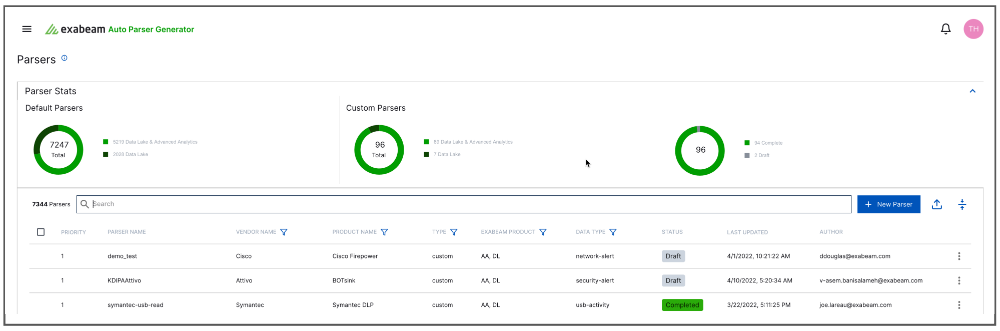
ALERT TRIAGE
Speed up your detection and investigation

AUTO PARSER GENERATOR
Easily create, edit, and check for new parsers

ACTION EDITOR
Create and edit custom actions packages

APG tool access in Exabeam

In the APG tool, click **New Parser**.

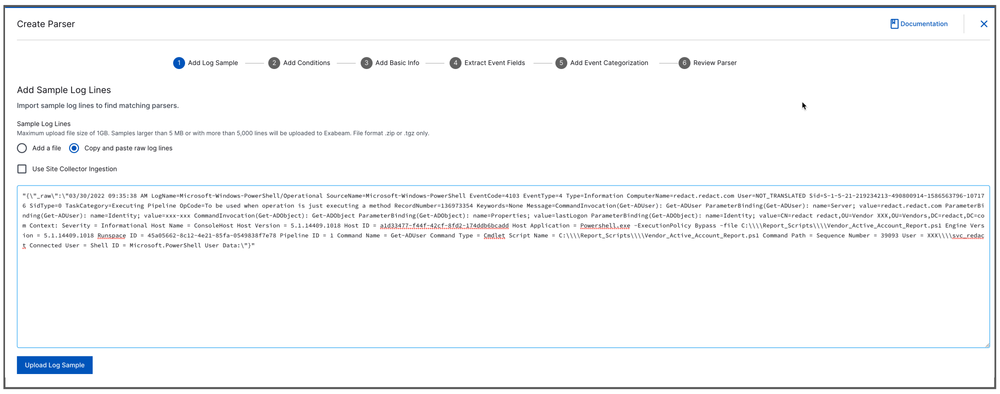


PRIORITY	PARSER NAME	VENDOR NAME	PRODUCT NAME	TYPE	EXABEAM PRODUCT	DATA TYPE	STATUS	LAST UPDATED	AUTHOR
1	demo_test	Cisco	Cisco Firepower	custom	AA, DL	network-alert	Draft	4/1/2022, 10:21:22 AM	ddouglas@exabeam.com
1	KDIPAAttivo	Attivo	BOTsink	custom	AA, DL	security-alert	Draft	4/10/2022, 5:20:34 AM	v-asem.banisalameh@exabeam.com
1	symantec-usb-read	Symantec	Symantec DLP	custom	AA, DL	usb-activity	Completed	3/22/2022, 5:11:25 PM	joe.lareau@exabeam.com

New Parser option

On the **Create Parser** page, click **Copy and paste raw log lines**.

In the text box, paste the **Message** field value from the your sample file and click **Upload Log Sample**.



Create Parser

Add Sample Log Lines

Import sample log lines to find matching parsers.

Sample Log Lines

Maximum upload file size of 1GB. Samples larger than 5 MB or with more than 5,000 lines will be uploaded to Exabeam. File format .zip or .tgz only.

( ) Add a file [✓ Copy and paste raw log lines]

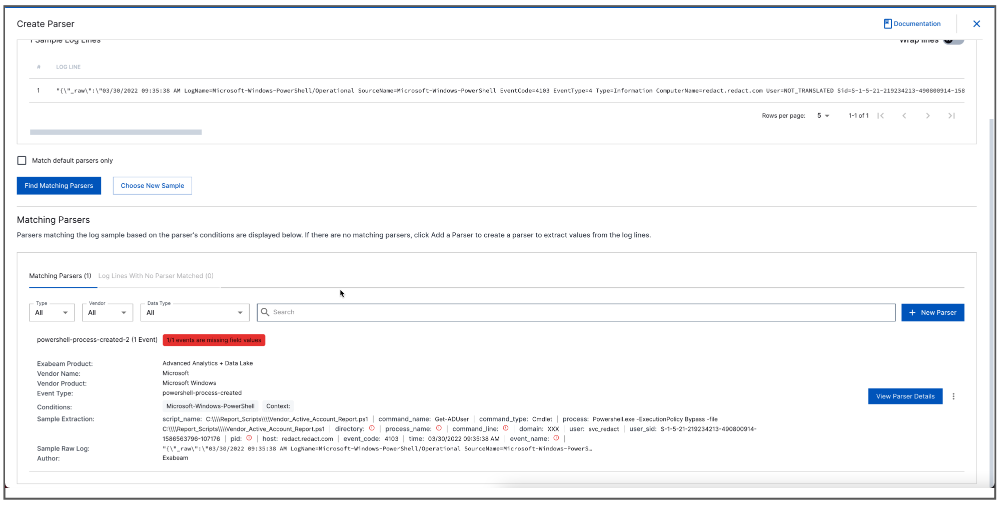
☐ Use Site Collector Ingestion

```
"{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=To be used when operation is just executing a method RecordNumber=136973354 Keywords=None Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding(Get-ADUser): name=Server; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject): Get-ADObject ParameterBinding(Get-ADObject): name=Properties; value=lastLogon ParameterBinding(Get-ADObject): name=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com Context: Severity = Informational Host Name = ConsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application = Powershell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet Script Name = C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Command Path = Sequence Number = 39093 User = XXX\\\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Data:\"}"
```

Paste Message Field

Copy the *Message** field value to your clipboard for a later step in Stream it to Exabeam.

Click **View Parsing Details** to view parsers matching this event type.



Create Parser

```
#  LOG LINE
1  "{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-158
```

Rows per page: 5 1-1 of 1

☐ Match default parsers only

Matching Parsers

Parsers matching the log sample based on the parser's conditions are displayed below. If there are no matching parsers, click Add a Parser to create a parser to extract values from the log lines.

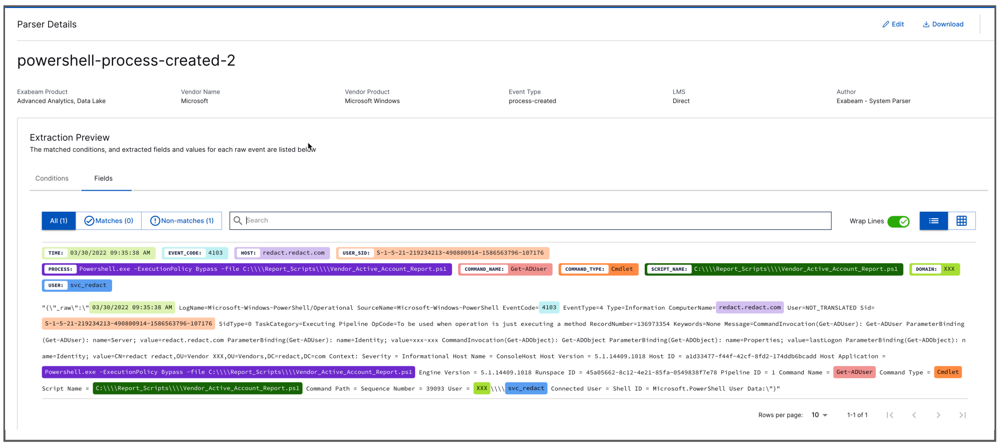
Type: All Vendor: All Data Type: All

powershell-process-created-2 (1 Event) 1/1 events are missing field values

Exabeam Product:	Advanced Analytics + Data Lake
Vendor Name:	Microsoft
Vendor Product:	Microsoft Windows
Event Type:	powershell-process-created
Conditions:	Microsoft-Windows-PowerShell Context:
Sample Extraction:	script_name: C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 | command_name: Get-ADUser | command_type: Cmdlet | process: Powershell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 | directory: ⓘ | process_name: ⓘ | command_line: ⓘ | domain: XXX | user: svc_redact | user_sid: S-1-5-21-219234213-490800914-1586563796-107176 | pid: ⓘ | host: redact.redact.com | event_code: 4103 | time: 03/30/2022 09:35:38 AM | event_name: ⓘ |
Sample Raw Log:	"{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerS…
Author:	Exabeam

New Parser option

Validate that all the fields you need are populated: `src_ip`, `dest_host`, `user`, etc.



Parser Details

powershell-process-created-2

Exabeam Product	Vendor Name	Vendor Product	Event Type	LMS	Author
Advanced Analytics, Data Lake	Microsoft	Microsoft Windows	process-created	Direct	Exabeam - System Parser

Extraction Preview

The matched conditions, and extracted fields and values for each raw event are listed below

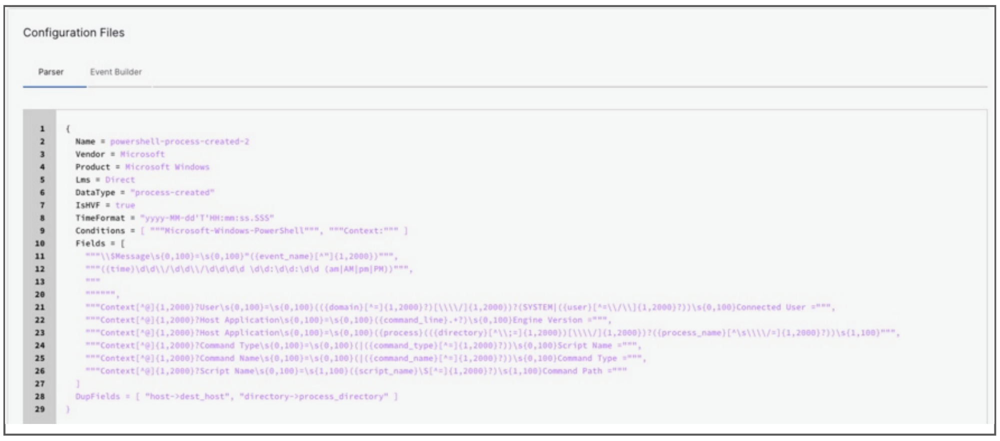
```
TIME: 03/30/2022 09:35:38 AM  EVENT_CODE: 4103  HOST: redact.redact.com  USER_SID: S-1-5-21-219234213-490800914-1586563796-107176
PROCESS: Powershell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1  COMMAND_NAME: Get-ADUser  COMMAND_TYPE: Cmdlet  SCRIPT_NAME: C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1  DOMAIN: XXX
USER: svc_redact

"{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=
S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=To be used when operation is just executing a method RecordNumber=136973354 Keywords=None Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding
(Get-ADUser): name=Server; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject): Get-ADObject ParameterBinding(Get-ADObject): name=Properties; value=lastLogon ParameterBinding(Get-ADObject): n
ame=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com Context: Severity = Informational Host Name = ConsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application =
Powershell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet
Script Name = C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Command Path = Sequence Number = 39093 User = XXX\\\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Data:\"}"
```

Rows per page: 10 1-1 of 1

Paste Message Field

Do not change Field Names in Exabeam (e.g., `src_ip` to `source_IP`). Exabeam has its own Field Name format that matches its Advanced Analytics template.

At the bottom of the **Extraction Preview** page, click **Configuration Files** to inspect the parsers. If required, you can download the parser to change the regex configurations.



```
{
  Name = powershell-process-created-2
  Vendor = Microsoft
  Product = Microsoft Windows
  Lms = Direct
  DataType = "process-created"
  IsHVF = true
  TimeFormat = "yyyy-MM-dd'T'HH:mm:ss.SSS"
  Conditions = [ """Microsoft-Windows-PowerShell""", """Context:""" ]
  Fields = [
    """\\$Message\s{0,100}=\s{0,100}"({event_name}[^"]{1,2000})""",
    """({time}\d\d\\/\d\d\\/\d\d\d\d \d\d:\d\d:\d\d (am|AM|pm|PM))""",
    """
    """""",
    """Context[^@]{1,2000}?User\s{0,100}=\s{0,100}((({domain}[^=]{1,2000}?)[\\\\/]{1,2000})?(SYSTEM|({user}[^=\\/\\]{1,2000}?))\s{0,100}Connected User =""",
    """Context[^@]{1,2000}?Host Application\s{0,100}=\s{0,100}({command_line}.+?)\s{0,100}Engine Version =""",
    """Context[^@]{1,2000}?Host Application\s{0,100}=\s{0,100}({process}(({directory}[^\\;=]{1,2000})[\\\\/]{1,2000})?({process_name}[^\s\\\\/=]{1,2000}?))\s{1,100}""",
    """Context[^@]{1,2000}?Command Type\s{0,100}=\s{0,100}(|({command_type}[^=]{1,2000}?))\s{0,100}Script Name =""",
    """Context[^@]{1,2000}?Command Name\s{0,100}=\s{0,100}(|({command_name}[^=]{1,2000}?))\s{0,100}Command Type =""",
    """Context[^@]{1,2000}?Script Name\s{0,100}=\s{1,100}({script_name}\S[^=]{1,2000}?)\s{1,100}Command Path ="""
  ]
  DupFields = [ "host->dest_host", "directory->process_directory" ]
}
```

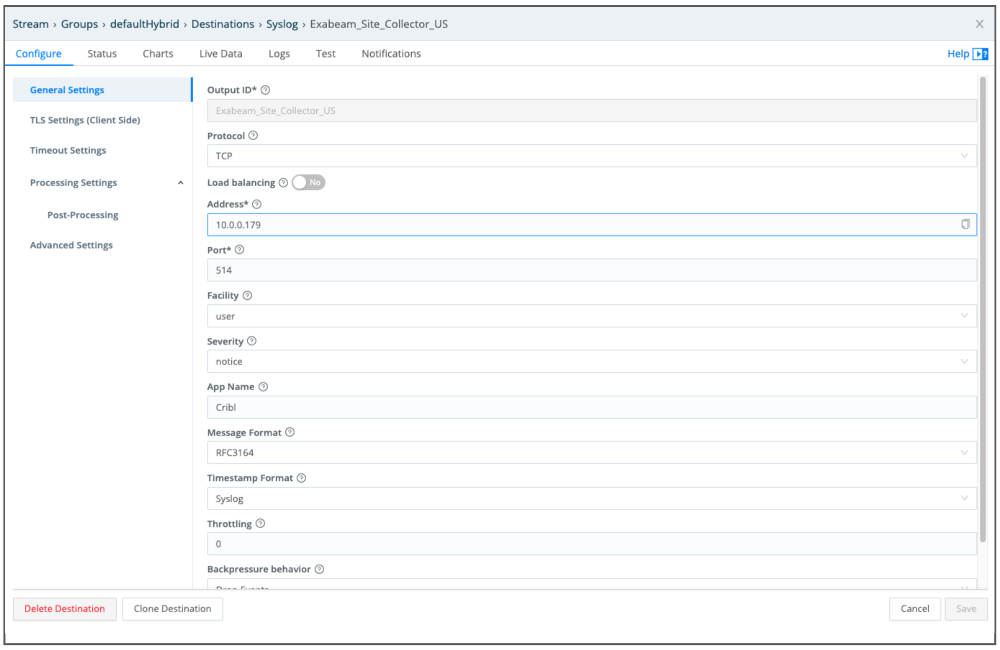
Parser Details

Stream It to Exabeam

Our final test is to send a single event to Exabeam, and validate the results in Exabeam's Data Lake or Advanced Analytics.

Before testing, configure your Exabeam implementation's `Asset Name` and `Username` to a dummy account. Exabeam's Advanced Analytics has hundreds of models out of the box, and you do not want to ruin these models while testing out your data.

1. From Cribl Stream's **Manage** submenu, select **Data** > **Destinations** > **Syslog**.
2. Click **Add Destination** to configure and save a new Syslog Destination to export data to Exabeam. Configure the **Address**, **Port**, **Message format**, **Timestamp format**, and other options to match your Exabeam implementation.


Stream › Groups › defaultHybrid › Destinations › Syslog › Exabeam_Site_Collector_US

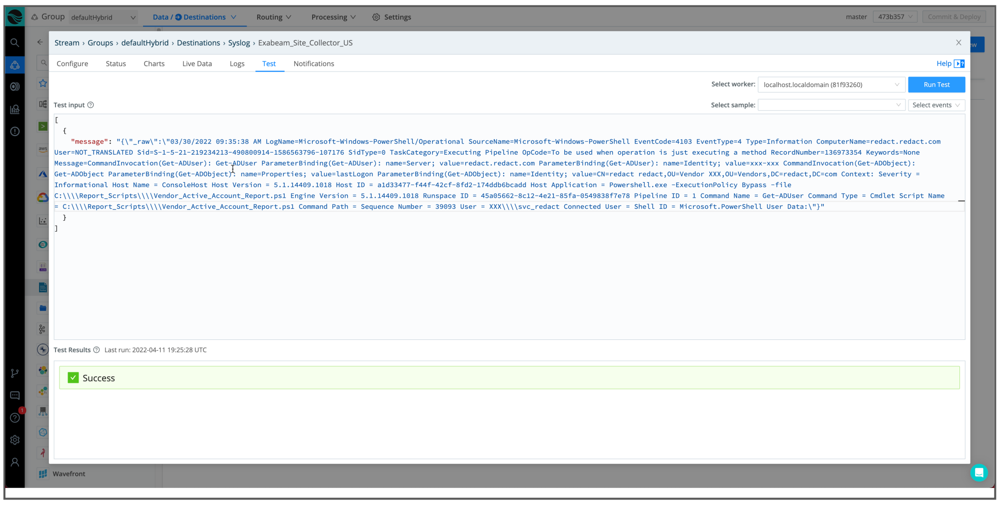
Configure | Status | Charts | Live Data | Logs | Test | Notifications

- General Settings
- TLS Settings (Client Side)
- Timeout Settings
- Processing Settings
  - Post-Processing
- Advanced Settings

Output ID*	Exabeam_Site_Collector_US
Protocol	TCP
Load balancing	No
Address*	10.0.0.179
Port*	514
Facility	user
Severity	notice
App Name	Cribl
Message Format	RFC3164
Timestamp Format	Syslog
Throttling	0
Backpressure behavior	

Configure Syslog

3. Reopen the Syslog Destination. On its **Test** tab, copy and paste the **Message** field's value you copied onto your clipboard above.


Stream › Groups › defaultHybrid › Destinations › Syslog › Exabeam_Site_Collector_US

Configure | Status | Charts | Live Data | Logs | Test | Notifications

Select worker: localhost.localdomain (81f93260)

Select sample:

Test input

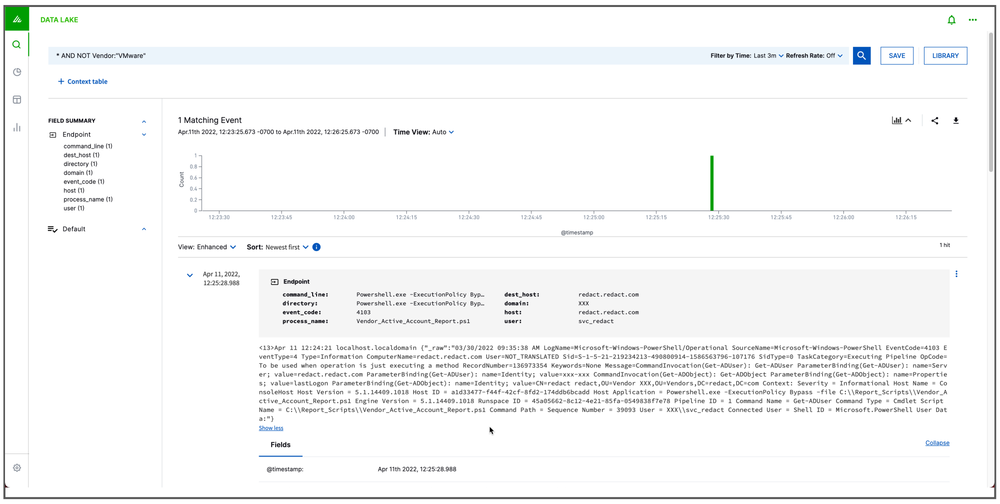
```
[
  {
    "message": "{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com
User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=To be used when operation is just executing a method RecordNumber=136973354 Keywords=None
Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding(Get-ADUser): name=Server; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject):
Get-ADObject ParameterBinding(Get-ADObject): name=Properties; value=lastLogon ParameterBinding(Get-ADObject): name=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com Context: Severity =
Informational Host Name = ConsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application = Powershell.exe -ExecutionPolicy Bypass -file
C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet Script Name
= C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1 Command Path = Sequence Number = 39093 User = XXX\\\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Data:\"}"
  }
]
```

Test Results   Last run: 2022-04-11 19:25:28 UTC

Success

Syslog Test tab

4. Within a few seconds, you should see your event display in the Exabeam Data Lake. Search for your forwarder IP/Host (Example syntax: `Forwarder:”IP/host`).



* AND NOT Vendor:"VMware"

Filter by Time: Last 3m Refresh Rate: Off

1 Matching Event

Apr.11th 2022, 12:23:25.673 -0700 to Apr.11th 2022, 12:26:25.673 -0700 | Time View: Auto

View: Enhanced Sort: Newest first

1 hit

Apr 11, 2022, 12:25:28.988

Endpoint

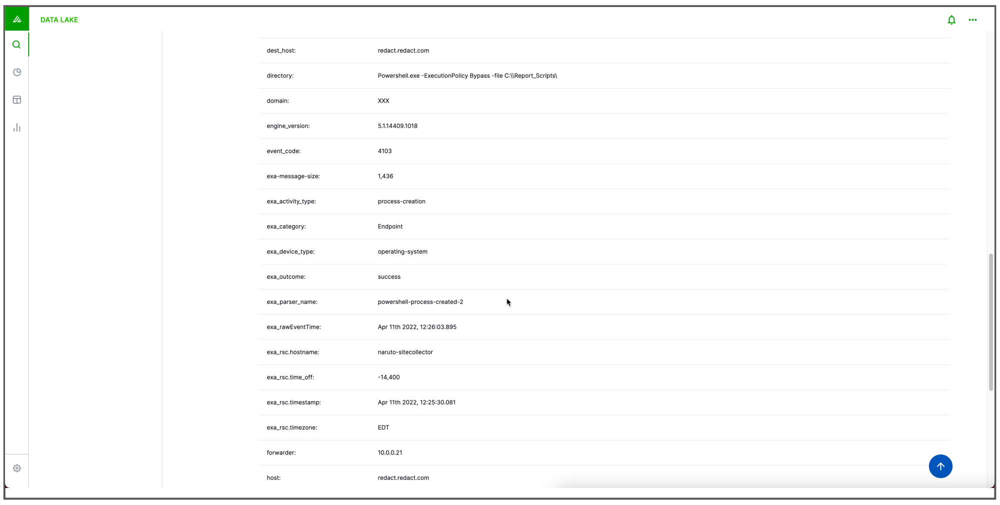
```
command_line:     Powershell.exe -ExecutionPolicy Byp…     dest_host:     redact.redact.com
directory:        Powershell.exe -ExecutionPolicy Byp…     domain:        XXX
event_code:       4103                                     host:          redact.redact.com
process_name:     Vendor_Active_Account_Report.ps1         user:          svc_redact
```

```
<13>Apr 11 12:24:21 localhost.localdomain {"_raw":"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 E
ventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 TaskCategory=Executing Pipeline OpCode=
To be used when operation is just executing a method RecordNumber=136973354 Keywords=None Message=CommandInvocation(Get-ADUser): Get-ADUser ParameterBinding(Get-ADUser): name=Serv
er; value=redact.redact.com ParameterBinding(Get-ADUser): name=Identity; value=xxx-xxx CommandInvocation(Get-ADObject): Get-ADObject ParameterBinding(Get-ADObject): name=Propertie
s; value=lastLogon ParameterBinding(Get-ADObject): name=Identity; value=CN=redact redact,OU=Vendor XXX,OU=Vendors,DC=redact,DC=com Context: Severity = Informational Host Name = Co
nsoleHost Host Version = 5.1.14409.1018 Host ID = a1d33477-f44f-42cf-8fd2-174ddb6bcadd Host Application = Powershell.exe -ExecutionPolicy Bypass -file C:\\Report_Scripts\\Vendor_A
ctive_Account_Report.ps1 Engine Version = 5.1.14409.1018 Runspace ID = 45a05662-8c12-4e21-85fa-0549838f7e78 Pipeline ID = 1 Command Name = Get-ADUser Command Type = Cmdlet Script
Name = C:\\Report_Scripts\\Vendor_Active_Account_Report.ps1 Command Path = Sequence Number = 39093 User = XXX\\svc_redact Connected User = Shell ID = Microsoft.PowerShell User Dat
a:"}
```

@timestamp:	Apr 11th 2022, 12:25:28.988

Validating event in Exabeam

5. You can validate that you have the right parser by matching the `exa_parser_name` with the parser in the Auto Parser Generator.



dest_host:	redact.redact.com
directory:	Powershell.exe -ExecutionPolicy Bypass -file C:\\Report_Scripts\
domain:	XXX
engine_version:	5.1.14409.1018
event_code:	4103
exa-message-size:	1,436
exa_activity_type:	process-creation
exa_category:	Endpoint
exa_device_type:	operating-system
exa_outcome:	success
exa_parser_name:	powershell-process-created-2
exa_rawEventTime:	Apr 11th 2022, 12:26:03.895
exa_rsc.hostname:	naruto-sitecollector
exa_rsc.time_off:	-14,400
exa_rsc.timestamp:	Apr 11th 2022, 12:25:30.081
exa_rsc.timezone:	EDT
forwarder:	10.0.0.21
host:	redact.redact.com

Validating parser in Exabeam

17.6.3.3. Splunk Stream To Cribl Stream

Splunk Stream is a set of three Splunk packages that, combined, enable you to capture and work with streams of network event data. This adds up to a collector for streaming data. You can deploy Splunk Stream in either of two forms:

- As part of a Splunk Universal Forwarder (we'll call this the forwarder-based config), sending data to a Cribl Stream Splunk TCP Source; or
- As an independent stream forwarder (we'll call this the ISF config), running on a compatible Linux machine, and sending data to a Cribl Stream Splunk HEC Source.

In either case, the collector process (not to be confused with a Cribl Stream Collector) will need to call home to a Splunk Enterprise process with the Splunk Stream app installed. You'll manage the collector's settings in Splunk Enterprise.

For the broader Splunk configuration story, see Splunk's documentation. Here, we'll explain one small part of the Splunk configuration process: how to configure Splunk Stream to send the data it captures to Cribl Stream. See the section below – either Forwarder-based or ISF – that corresponds to your use case.

Setting Up the Forwarder-Based Config

Install the Stream TA on the Universal Forwarder targets, either manually or via the deployment server. You'll need to add a new `inputs.conf` stanza pointing to your Splunk Stream App management instance. Adapt the example stanza below, replacing the placeholder with the hostname or IP address of your management host.

New stanza in config file

```
[streamfwd://streamfwd]
splunk_stream_app_location = https://<your_management_host>:8000/en-us/custom/splu
disabled = 0
```

The `outputs.conf` for the Universal Forwarder with Splunk Stream App is the same as for a Universal Forwarder with a non-streaming collector. See these examples.

To verify that your setup is working, run a Live Capture in your Cribl Stream Splunk TCP Source with appropriate filters. Once the Universal Forwarder is sending data to your Cribl Stream Workers, you're ready to begin working with sample captures, Routes, and Pipelines.

If no data seems to be coming through, check the logs located at `/opt/splunkforwarder/var/log/splunk/splunkd.log` on each machine where your Forwarder is running.

Setting Up the ISF Config

Splunk Stream as an independent stream forwarder (ISF) can run only on Ubuntu- or RHEL-based x64 Linux machines that have bzip2 installed.

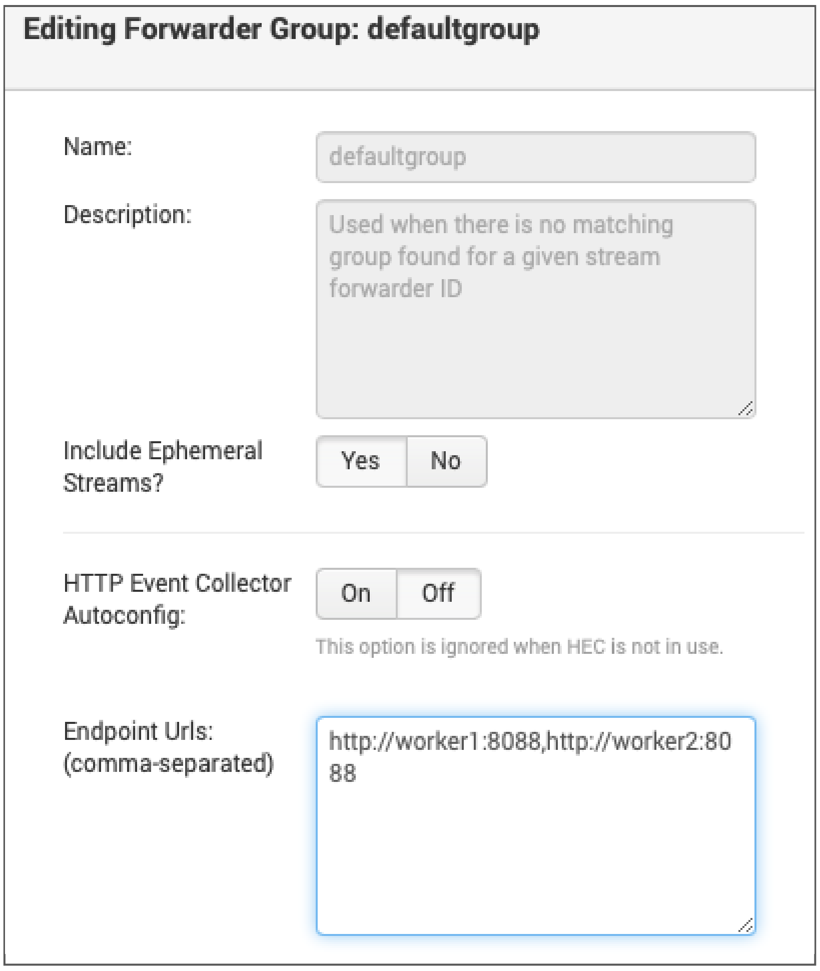
To install Splunk Stream, begin in the Splunk UI:

- In the Stream App, navigate to **Configuration** > **Distributed Forwarder Management**.
- Click the **Install Stream Forwarders** button.
- In the resulting modal, under the text `To get data from other machines, run this command on your data source machine`, copy the `curl` command. E.g.:
  ```
  curl -sSL http://DellT20:8000/en-us/custom/splunk_app_stream/install_streamfwd
  | sudo bash
  ```

On each Linux machine where you want to install Splunk Stream, run the `curl` command that you copied. Installation will fail if the machine lacks `bzip2`.

Once installation is complete, return to the Splunk UI.

1. To update the HEC endpoint URL, navigate to **Actions** > **Edit Forwarder Group** for the desired group.
2. Toggle **HTTP Event Collector Autoconfig** to `Off`.
3. In the **Endpoint URLs** field, list your Cribl Stream Worker URLs, as shown here:


Editing Forwarder Group: defaultgroup

Name: defaultgroup

Description: Used when there is no matching group found for a given stream forwarder ID

Include Ephemeral Streams? Yes No

HTTP Event Collector Autoconfig: On Off

This option is ignored when HEC is not in use.

Endpoint Urls: (comma-separated) http://worker1:8088,http://worker2:8088

Editing forwarder group to specify Cribl Worker URLs

4. Click **OK**.

Finally, on each Linux machine where you're running Splunk Stream, update the ISF settings with the proper HEC token. (While it is technically possible to run without a token, Cribl strongly recommends against this practice.)

1. In Cribl Stream, copy the HEC token from your Cribl Stream Splunk HEC Source definition (the value of the `__hecToken` internal field).
2. On the desired Linux machines, open `/opt/streamfwd/local/streamfwd.conf` in a text editor.
3. Add the following stanza, substituting your HEC token for the placeholder:

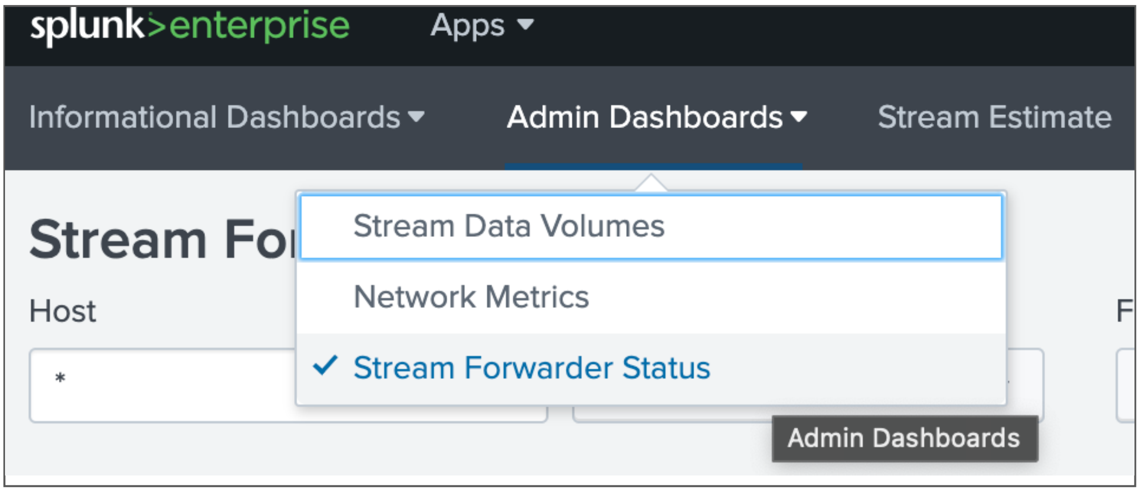
```
[streamfwd]
httpEventCollectorToken = <HEC_token_from_your_Cribl_Stream_Source>
```

4. Run the following command to restart the ISF:

```
systemctl restart streamfwd
```

To verify that your setup is working, run a Live Capture in your Cribl Stream Splunk HEC Source with appropriate filters.

You can also check ISF status in Splunk. Navigate to **Apps** > **Splunk Stream** > **Admin Dashboards** > **Stream Forwarder Status**:



Verify ISF status in Splunk

If no data seems to be coming through, check the logs located at `/opt/streamfwd/var/log/streamfwd.log` on each machine where your ISF is running.

17.6.3.4. Splunk Cloud And BYOL Integrations

Cribl Stream can send data to these flavors of Splunk Cloud:

- The free, single-instance trial version.
- A distributed Splunk Cloud instance with clustered indexers.
- A Bring Your Own License (BYOL) deployment, either in a non-Splunk cloud or on-prem.

You have a choice of two methods for sending the data:

- Splunk HEC (HTTP Event Collector).
- The S2S (Splunk-to-Splunk) protocol.

Of all the possible combinations, three have proven most useful in the field:

- Using Splunk HEC with the trial version of Splunk.
- Using S2S with a distributed instance of Splunk.
- Using S2S with a BYOL deployment of Splunk.

> 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.

When to Use Splunk HEC

Splunk HEC is fast and easy to set up. Under the hood, it uses the HTTP/S protocol. This offers better compression than S2S, which is a binary protocol.

The Splunk HEC endpoints are virtual endpoints, front-ended with load balancers – ELB for AWS, or GLB for GCP. This provides good load-balancing.

Cribl generally recommends using Splunk HEC for integrating with Splunk Cloud, because (1) it requires fewer connections than S2S, and therefore consumes less memory; and (2) because its superior compression yields lower egress costs.

When to Use S2S

S2S allows each Cribl Stream Worker Process to connect to multiple indexers concurrently, which distributes data very effectively. This helps significantly with Splunk search, by placing a smaller burden on a larger number of indexers. This support for concurrent connections is the main advantage of S2S. Consider S2S if you plan to route all your data through Cribl Stream first, and you prioritize search performance.

Using Splunk HEC

Identify Your Splunk HEC Endpoint

In Splunk Cloud, identify your HEC endpoint, as described in the Splunk documentation. Here are some example URL patterns for HEC endpoints:

- Free version: `https://inputs.<cloud_stack_name>:8088/<endpoint>`
- Paid Version in AWS: `https://http-inputs-<cloud_stack_name>:443/<endpoint>`
- Paid version in GCP: `https://http-inputs.<cloud_stack_name>:443/<endpoint>`

A HEC endpoint for a paid version of Splunk Cloud on AWS with an endpoint for JSON-formatted events, for a company called "Acme Group," might look like this:

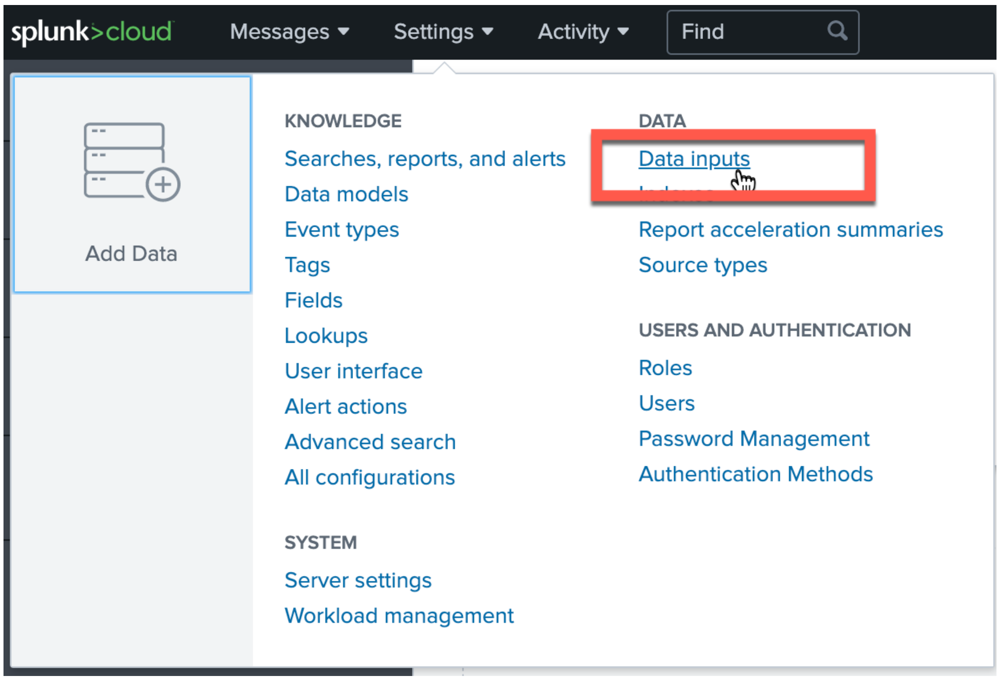
```
https://http-inputs-acmegroup.splunkcloud.com:443/services/collector/event
```

Copy the endpoint URL for use when configuring Cribl Stream in the next section.

Create HEC Tokens

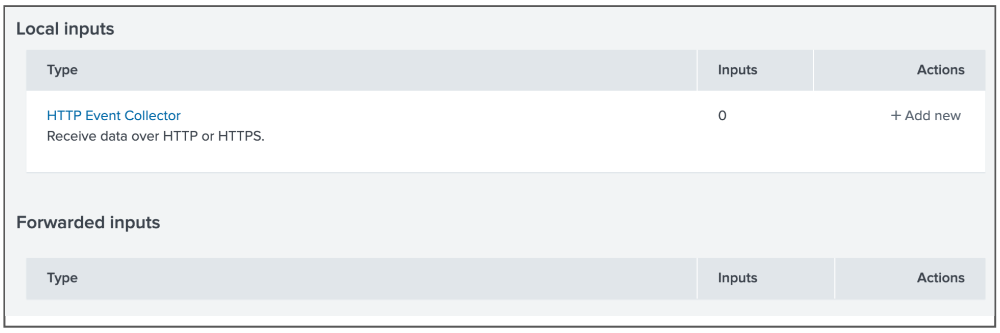
You need to create at least one HEC token. For deployments where you set up routing to individual indexes, or you use HEC tokens for RBAC on Splunk, you will create multiple HEC tokens.

1. In the Splunk UI, open the **Settings** menu and click **Data Inputs**.



Settings > Data inputs

2. In the resulting modal's **HTTP Event Collector** section, click **+ Add new**.



Local inputs

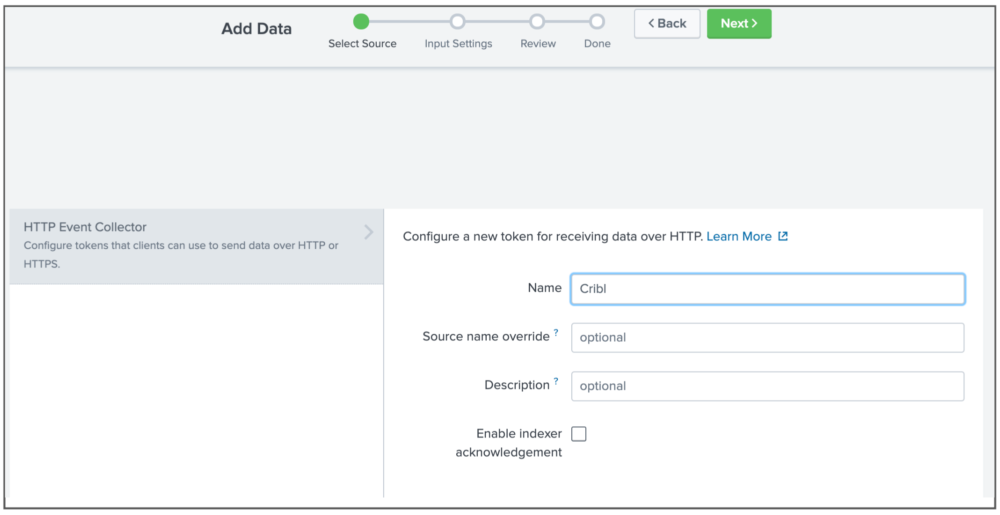
Type	Inputs	Actions
HTTP Event Collector Receive data over HTTP or HTTPS.	0	+ Add new

Forwarded inputs

Type	Inputs	Actions

Adding a new HTTP Event Collector

3. Name the new token and click **Next**.



Configure a new token for receiving data over HTTP. Learn More

Name: Cribl

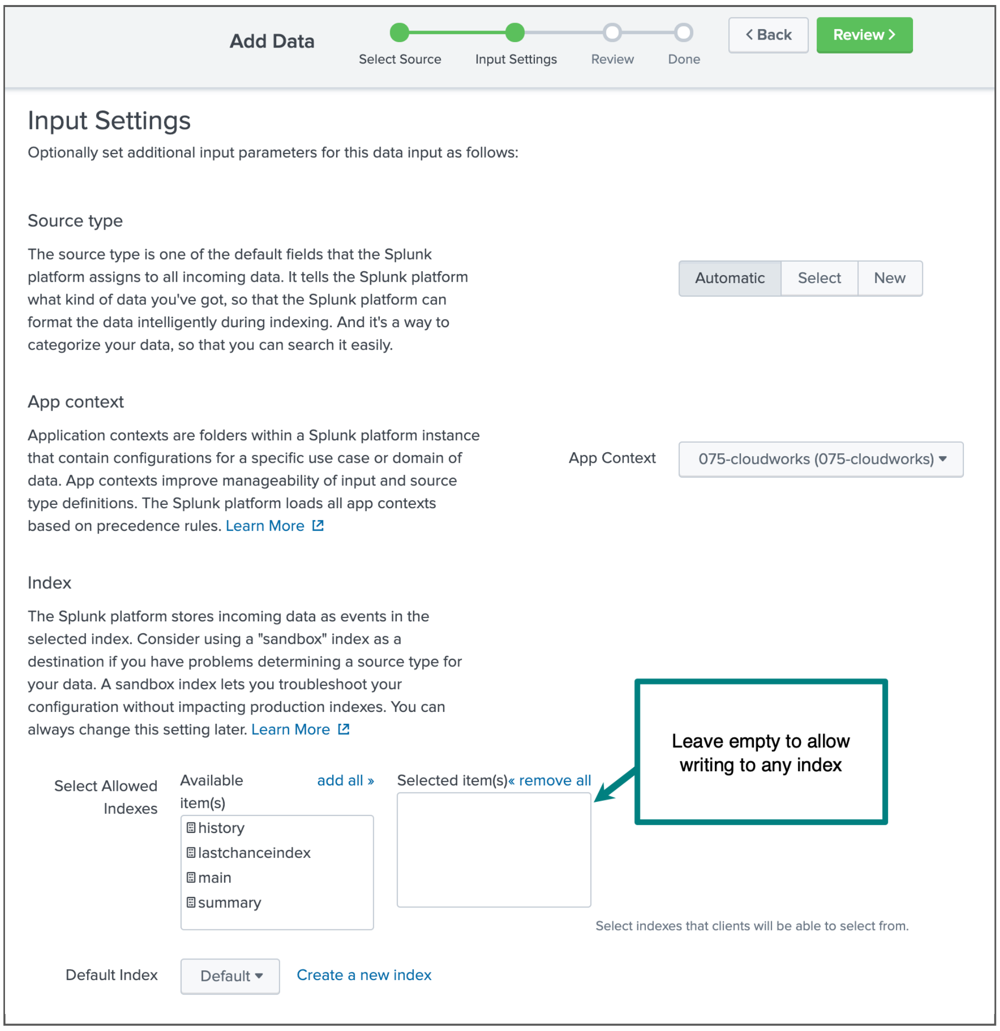
Source name override: optional

Description: optional

Enable indexer acknowledgement: ☐

Adding a new token

4. Do not add any indexes. This way HEC can write to any index. If you prefer a default index other than `main`, choose it from the **Default index** drop-down.



Add Data

Select Source · Input Settings · Review · Done

< Back · Review >

Input Settings

Optionally set additional input parameters for this data input as follows:

Source type

The source type is one of the default fields that the Splunk platform assigns to all incoming data. It tells the Splunk platform what kind of data you've got, so that the Splunk platform can format the data intelligently during indexing. And it's a way to categorize your data, so that you can search it easily.

Automatic · Select · New

App context

Application contexts are folders within a Splunk platform instance that contain configurations for a specific use case or domain of data. App contexts improve manageability of input and source type definitions. The Splunk platform loads all app contexts based on precedence rules. Learn More

App Context: 075-cloudworks (075-cloudworks)

Index

The Splunk platform stores incoming data as events in the selected index. Consider using a "sandbox" index as a destination if you have problems determining a source type for your data. A sandbox index lets you troubleshoot your configuration without impacting production indexes. You can always change this setting later. Learn More

Select Allowed Indexes

Available item(s) — add all »
- history
- lastchanceindex
- main
- summary

Selected item(s) — « remove all

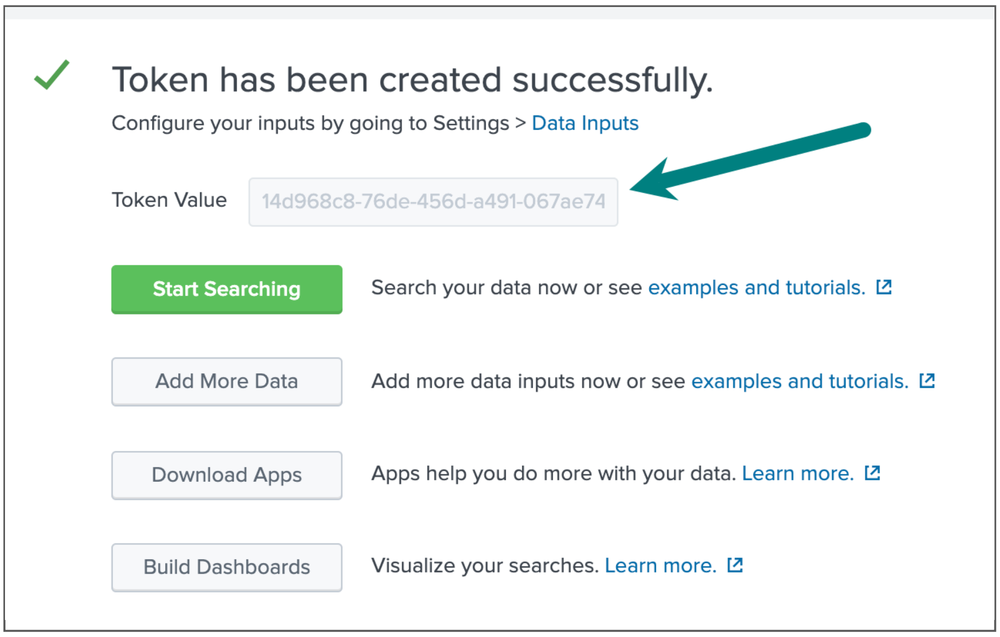
Leave empty to allow writing to any index

Select indexes that clients will be able to select from.

Default Index: Default · Create a new index

Indexes for the new token

5. Once the token has been created, copy it for use when configuring Cribl Stream in the next section.



Token has been created successfully.

Configure your inputs by going to Settings > Data Inputs

Token Value: 14d968c8-76de-456d-a491-067ae74

Start Searching — Search your data now or see examples and tutorials.

Add More Data — Add more data inputs now or see examples and tutorials.

Download Apps — Apps help you do more with your data. Learn more.

Build Dashboards — Visualize your searches. Learn more.

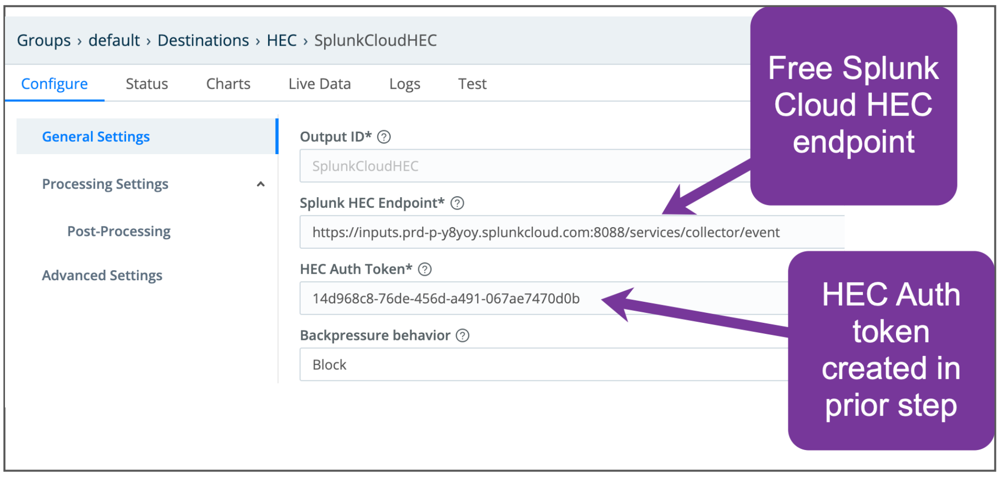
Copying the token value

Add a Splunk HEC Destination in Cribl Stream

From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Destinations**, then select **Splunk** > **HEC** from the **Manage Destinations** page's tiles or left nav. Click **Add New** to open the **HEC** > **New Destination** modal.

In the **General Settings** tab:

- Grab the values that you copied in the previous section, and paste them into the **Splunk HEC Endpoint** and **HEC Auth Token** fields, respectively. Be sure to specify HTTPS, because the endpoint will default to HTTP.
- Click **Save**.
- Click **Commit & Deploy**.


Groups › default › Destinations › HEC › SplunkCloudHEC
Configure
Status
Charts
Live Data
Logs
Test
General Settings
Processing Settings
Post-Processing
Advanced Settings
Free Splunk Cloud HEC endpoint
HEC Auth token created in prior step

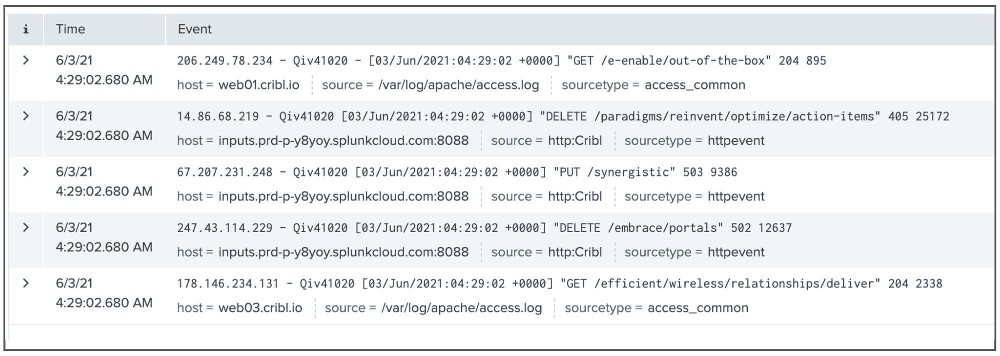
Output ID* SplunkCloudHEC
Splunk HEC Endpoint* https://inputs.prd-p-y8yoy.splunkcloud.com:8088/services/collector/event
HEC Auth Token* 14d968c8-76de-456d-a491-067ae7470d0b
Backpressure behavior Block

Populating General Settings with values from Splunk

Verify that Data is Flowing from Cribl Stream to Splunk Cloud

Be sure you have committed and deployed the newly created configuration. Otherwise, data will not flow to Splunk Cloud, and verification will fail.

1. In Cribl Stream, open the Splunk HEC Destination that you created in the previous section.
   - In the configuration modal's **Test** tab, click **Run Test**.
   - You should see a **Success** message.
2. In Splunk, search on `index=main cribl_pipe=*`. Events that you sent from the Cribl Stream **Test** tab should appear in the search results.



i	Time	Event

>	6/3/21 4:29:02.680 AM	`206.249.78.234 - Qiv41020 - [03/Jun/2021:04:29:02 +0000] "GET /e-enable/out-of-the-box" 204 895`
		host = web01.cribl.io source = /var/log/apache/access.log sourcetype = access_common

>	6/3/21 4:29:02.680 AM	`14.86.68.219 - Qiv41020 [03/Jun/2021:04:29:02 +0000] "DELETE /paradigms/reinvent/optimize/action-items" 405 25172`
		host = inputs.prd-p-y8yoy.splunkcloud.com:8088 source = http:Cribl sourcetype = httpevent

>	6/3/21 4:29:02.680 AM	`67.207.231.248 - Qiv41020 [03/Jun/2021:04:29:02 +0000] "PUT /synergistic" 503 9386`
		host = inputs.prd-p-y8yoy.splunkcloud.com:8088 source = http:Cribl sourcetype = httpevent

>	6/3/21 4:29:02.680 AM	`247.43.114.229 - Qiv41020 [03/Jun/2021:04:29:02 +0000] "DELETE /embrace/portals" 502 12637`
		host = inputs.prd-p-y8yoy.splunkcloud.com:8088 source = http:Cribl sourcetype = httpevent

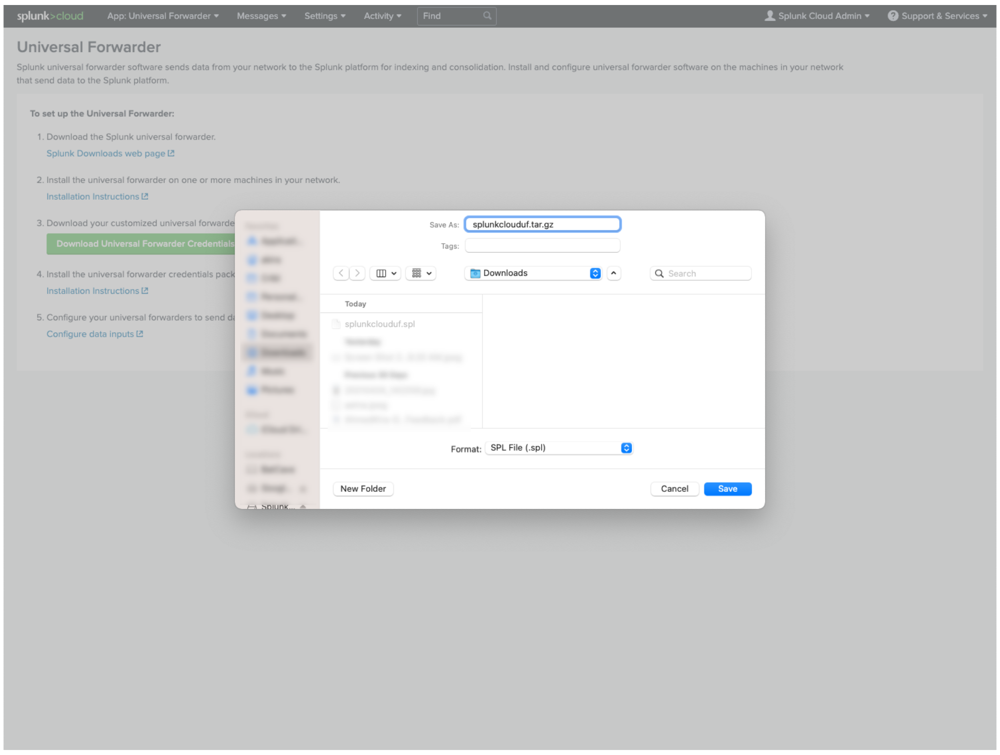
>	6/3/21 4:29:02.680 AM	`178.146.234.131 - Qiv41020 [03/Jun/2021:04:29:02 +0000] "GET /efficient/wireless/relationships/deliver" 204 2338`
		host = web03.cribl.io source = /var/log/apache/access.log sourcetype = access_common

Events flowing to Splunk

Using S2S

Prepare Splunk Cloud for Cribl Stream Integration

1. In Splunk Cloud, download the Splunk Cloud Universal Forwarder credentials app to your desktop.
2. Change the file suffix from `.spl` to `.tar.gz`.



Universal Forwarder

Splunk universal forwarder software sends data from your network to the Splunk platform for indexing and consolidation. Install and configure universal forwarder software on the machines in your network that send data to the Splunk platform.

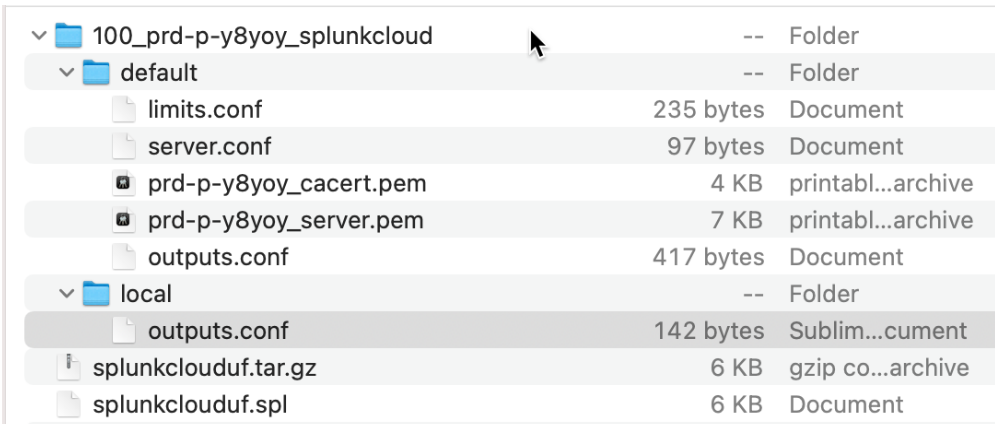
To set up the Universal Forwarder:

1. Download the Splunk universal forwarder.
   Splunk Downloads web page
2. Install the universal forwarder on one or more machines in your network.
   Installation Instructions
3. Download your customized universal forwarde
4. Install the universal forwarder credentials pack
   Installation Instructions
5. Configure your universal forwarders to send d
   Configure data inputs

Save As: splunkclouduf.tar.gz
Tags:
Downloads
Today
splunkclouduf.spl
Format: SPL File (.spl)

Changing the credentials app file suffix

3. Untar/unzip the directory to expose the files.



100_prd-p-y8yoy_splunkcloud	--	Folder
default	--	Folder
limits.conf	235 bytes	Document
server.conf	97 bytes	Document
prd-p-y8yoy_cacert.pem	4 KB	printabl...archive
prd-p-y8yoy_server.pem	7 KB	printabl...archive
outputs.conf	417 bytes	Document
local	--	Folder
outputs.conf	142 bytes	Sublim...cument
splunkclouduf.tar.gz	6 KB	gzip co...archive
splunkclouduf.spl	6 KB	Document

Credentials app files

4. Locate the following files. You will need them when you configure Cribl Stream in the next section.
   - `./default/<SplunkCloudInstanceName>_cacert.pem`
   - `./default/<SplunkCloudInstanceName>_server.pem`
   - `./default/outputs.conf`
   - `./local/outputs.conf`

Configure Certificate Settings in Cribl Stream

1. In the top menu, go to **Setting** > **Global Settings**.
2. Next, in the left nav, select **Security** > **Certificates**.
3. Populate each field below with the specified content:
   - **Certificate**: Drag and drop the `server.pem` file.
   - **Private Key**: Copy and paste just the `private key` section of the `server.pem` file.
   - **Passphrase**: Copy and paste just the SSL password from the `../local/outputs.conf` file.
   - **CA certificate**: Drag and drop the `cacert.pem` file.

Add a Splunk Destination in Cribl Stream

The type of Destination to add depends on what form of Splunk you're using:

- For a trial version of Splunk Cloud, select **Splunk Single Instance**.
- For a paid version of Splunk Cloud, select **Splunk Load Balanced**. This is required because any paid version of Splunk Cloud will have multiple indexer entries in the `../default/outputs.conf` file.

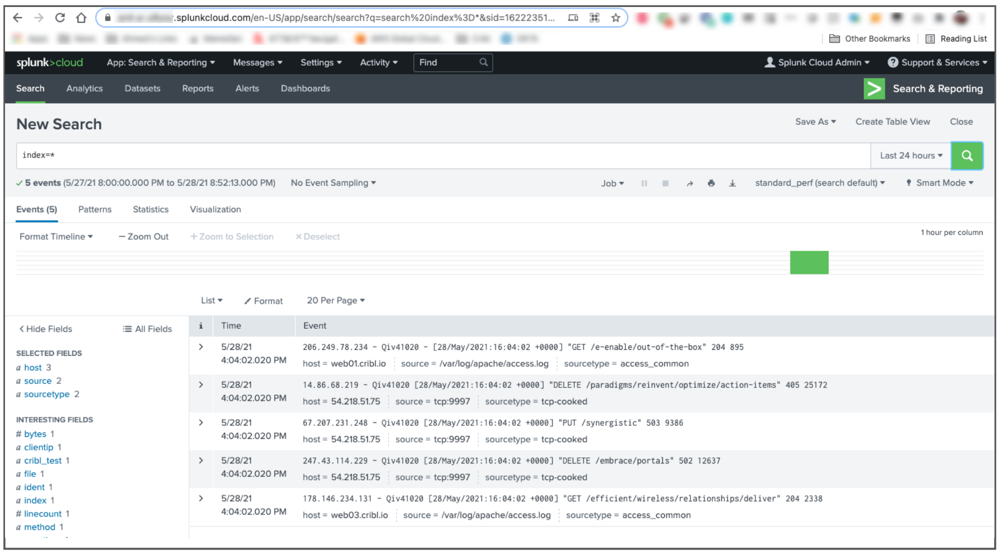
1. From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Destinations**, then select either **Splunk** > **Load Balanced** or **Splunk** > **Single Instance** from the **Manage Destinations** page's tiles or left nav. Then click **New Destination** to open the corresponding **New Destination** modal.

2. In the **General Settings** tab, populate the **Address** and **Port** fields.
   - From the `./default/outputs.conf` file you copied in the previous section, divide the value of the `server` line between the two fields shown in the screenshot below.
3. In the **TLS Settings (Client Side)** tab:
   - From the **Certificate name** drop-down, select the certificate that you created.
   - From the `./local/outputs.conf` file, paste the `sslPassword` value into the **Passphrase** field.
   - Click **Save**.
   - Click **Commit** and **Deploy**.

Verify that Data is Flowing from Cribl Stream to Splunk Cloud

Be sure you have committed and deployed the newly created configuration. Otherwise, data will not flow to Splunk Cloud, and verification will fail.

1. In Cribl Stream, open the Destination that you created in the previous section.
   - In the configuration modal's **Test** tab, click **Run Test**.
   - You should see a **Success** message.
2. In Splunk, search on `index=main cribl_pipe=*`. Events that you sent from the Cribl Stream Test tab should appear in the search results.



```
i  Time                      Event
>  5/28/21 4:04:02.020 PM    206.249.78.234 - Qiv41020 - [28/May/2021:16:04:02 +0000] "GET /e-enable/out-of-the-box" 204 895
                             host = web01.cribl.io   source = /var/log/apache/access.log   sourcetype = access_common
>  5/28/21 4:04:02.020 PM    14.86.68.219 - Qiv41020 [28/May/2021:16:04:02 +0000] "DELETE /paradigms/reinvent/optimize/action-items" 405 25172
                             host = 54.218.51.75   source = tcp:9997   sourcetype = tcp-cooked
>  5/28/21 4:04:02.020 PM    67.207.231.248 - Qiv41020 [28/May/2021:16:04:02 +0000] "PUT /synergistic" 503 9386
                             host = 54.218.51.75   source = tcp:9997   sourcetype = tcp-cooked
>  5/28/21 4:04:02.020 PM    247.43.114.229 - Qiv41020 [28/May/2021:16:04:02 +0000] "DELETE /embrace/portals" 502 12637
                             host = 54.218.51.75   source = tcp:9997   sourcetype = tcp-cooked
>  5/28/21 4:04:02.020 PM    178.146.234.131 - Qiv41020 [28/May/2021:16:04:02 +0000] "GET /efficient/wireless/relationships/deliver" 204 2338
                             host = web03.cribl.io   source = /var/log/apache/access.log   sourcetype = access_common
```

Events flowing to Splunk

Using S2S with Splunk BYOL

Before you begin configuring this option, you should already have Splunk Universal Forwarders configured to send data securely to your Splunk environment. This enables you to:

- Re-use content from the `.pem` and `outputs.conf` files already in use on those Forwarders.
- Follow the procedures in the in the previous section to add the certificate to Cribl.Cloud.
- Then, reference the certificate in the Splunk Destination configuration.

If you need to secure your Splunk indexers, see the Splunk documentation.

When Your Data Source is a Splunk Forwarder

If a Splunk Universal or Heavy Forwarder is the source of the data you want to send to Splunk Cloud:

- In Cribl Stream, create a Splunk TCP Source to receive data from the Splunk Forwarder.
- This process includes configuring the Splunk Forwarder to point to the new Source in Cribl Stream, and (optionally) securing the communication with TLS.

17.6.4. Syslog

17.6.4.1. Syslog Best Practices

Cribl Stream can process a syslog stream directly. Moving to Cribl Stream from existing syslog-ng or rsyslog servers fully replaces those solutions with one that is fully supported and easily managed.

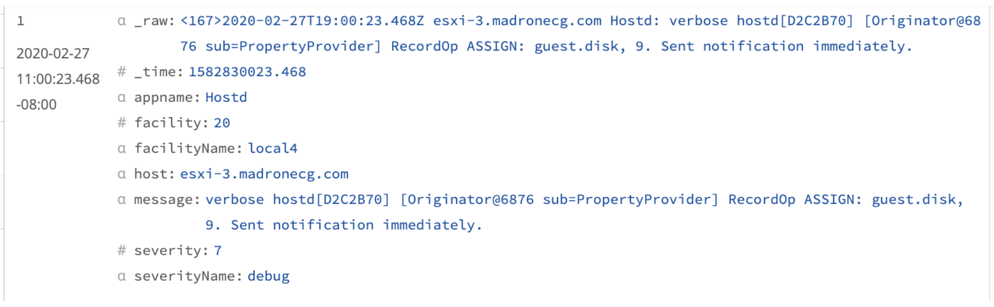
Processing syslog in Cribl Stream allows you to readily address these common challenges of ingesting data from syslog senders:

- **Architecture**: Cribl Stream routes the syslog stream directly, immediately, and securely to the destinations of your choice, reducing latency and management overhead.
- **Volume**: Removing redundant data and unnecessary fields in Cribl Stream typically reduces volume 20–30% overall. It also optimizes the data for downstream services like Splunk or Elasticsearch.
- **Timestamp handling**: Cribl Stream intelligently processes events sent from different time zones. It can embed a new, consistent timestamp, and can auto-correct timestamps that are off by an exact number of hours.
- **Severity/Facility accuracy**: Each syslog event begins with a bracketed integer that represents Facility and Severity, as defined in the Syslog Protocol. Cribl Stream translates this code (e.g., `<165>`, `<0>`) into the correct Facility and Severity values.
- **Metadata**: Cribl Stream can automatically set metadata fields, including `sourcetype` and `index`.

This tutorial outlines best practices for replacing your syslog server with Cribl Stream. To go even a bit deeper, check out this Cribl Office Hours video.

The Goal: Optimizing Syslog Events

By default, a Cribl Stream Syslog Source produces eight fields: `_time`, `appname`, `facility` (numeric), `facilityName` (text), `host`, `message`, `severity` (numeric), and `severityName` (text).

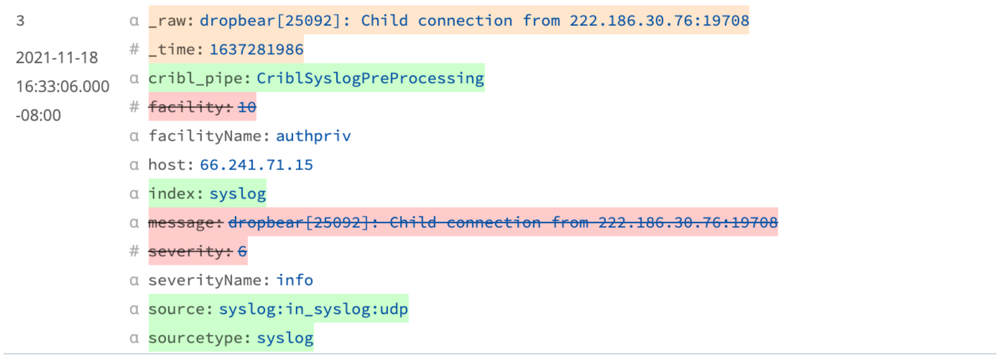


```
1               α _raw: <167>2020-02-27T19:00:23.468Z esxi-3.madronecg.com Hostd: verbose hostd[D2C2B70] [Originator@68
                        76 sub=PropertyProvider] RecordOp ASSIGN: guest.disk, 9. Sent notification immediately.
2020-02-27      # _time: 1582830023.468
11:00:23.468    α appname: Hostd
-08:00          # facility: 20
                α facilityName: local4
                α host: esxi-3.madronecg.com
                α message: verbose hostd[D2C2B70] [Originator@6876 sub=PropertyProvider] RecordOp ASSIGN: guest.disk,
                           9. Sent notification immediately.
                # severity: 7
                α severityName: debug
```

Default, parsed syslog event

While this default parsing makes the output much more readable, we haven't saved any volume – and we now have redundant pairs of fields (numeric versus text) that represent facility and severity.

Our next logical step is to streamline syslog events to something more like this:



```
3                 α _raw: dropbear[25092]: Child connection from 222.186.30.76:19708
2021-11-18        # _time: 1637281986
16:33:06.000      α cribl_pipe: CriblSyslogPreProcessing
-08:00            # ~~facility: 10~~
                  α facilityName: authpriv
                  α host: 66.241.71.15
                  α index: syslog
                  α ~~message: dropbear[25092]: Child connection from 222.186.30.76:19708~~
                  # ~~severity: 6~~
                  α severityName: info
                  α source: syslog:in_syslog:udp
                  α sourcetype: syslog
```

Default, parsed syslog event

This accomplishes all of the following:

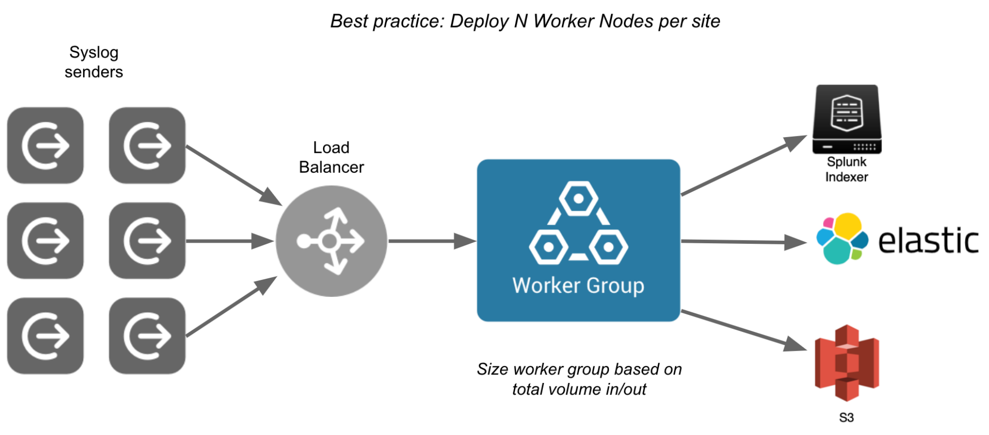
- Extracts the essentials.
- Removes the redundancies.
- Adds a new field to identify the Cribl Stream Pipeline (which we're about to build).
- Adds metadata that the Destination needs.
- Shrinks the outbound `_raw` payload to just its `message` component.

Once we optimize syslog in this way, we can achieve still further efficiencies by dropping or downsampling frequent events, and by balancing high-volume syslog inputs across Cribl Stream worker processes.

Overall Architecture

Syslog data, especially when sent via UDP, is best collected as close to the source as possible. Ideally, you should capture syslog data **at its origin**. (This is true of syslog in general, not just syslog processed in Cribl Stream.)

Also, because syslog senders have no built-in load balancing, Cribl strongly recommends using a load balancer to distribute the load across multiple Worker Nodes.


Best practice: Deploy N Worker Nodes per site
Syslog senders
Load Balancer
Worker Group
Size worker group based on total volume in/out
Splunk Indexer
elastic
S3

Example architecture for a single site or location

The load balancer shown above is dedicated only to communication between the syslog senders and the Stream Worker Nodes. Any load balancing between the Worker Group and its Stream Leader would be handled by a separate load balancer.

Load Balancing

When configuring a load balancer to be fed by syslog senders, start with these basic principles:

Avoid Stickiness

In this context, **stickiness** is when traffic from a given source is always sent to the same destination IP. The optimal configuration of the load balancer avoids "sticky" behavior, and instead spreads the workload across Cribl Stream Worker Nodes as evenly as possible.

Use API Calls to Do Health Checks

If UDP data is being sent, the load balancer has no way to automatically detect whether the destination is up. Configure the load balancer to use API calls to the Worker Nodes to check the health status of each Node (see Health Endpoint).

Listen on Port 514

If possible, configure the load balancer to listen on port 514, and then relay traffic to the Worker Nodes on port 9514. Here's why:

- Many syslog senders are hard-coded to send only to port 514, so you need to support them.

- Cribl Stream itself should not run as root, and therefore cannot listen on ports lower than 1024 without additional OS-level steps (like SETCAP).

A Port for Each Device Class

As explained above, for syslog devices that can send only to port 514, you can configure the load balancer to relay to destination port 9514. But what about the many syslog senders that **do** support sending to ports other than 514? It's a best practice among Cribl customers to use a dedicated receiving port for each class of device in their environment. While this takes a little more effort to set up, it enables you to hard-code metadata (and optionally, Pipelines) for the events from that data source. Examples include:

- 1517: VMware ESXi logs.
- 1521: Palo Alto Firewall.
- 1522: F5 load balancers.

We'll see how to set per-port metadata in the Adding Processing Pipelines section.

UDP Versus TCP

For any given syslog device, you might need to choose between using UDP or TCP. Which is best depends on the characteristics of the sender. Here are some basic guidelines:

- For single, high-volume senders (over 300GB/day), use UDP if possible. For both the sender and Cribl Stream, UDP imposes lower overhead than TCP. The stateless nature of UDP allows each log event to be directed to a different worker thread than the last. This ensures maximum utilization of the Worker Nodes. See Sizing and Scaling for more details.
- For lower-volume senders, use TCP if the sender supports it.
- For all other use cases, use UDP.

Pipeline Planning

In Cribl Stream, we speak of **pre-processing Pipelines**, **processing Pipelines** (or just plain **Pipelines**), and **post-processing Pipelines**. Cribl recommends combining the first two in an arrangement that we've found to be optimal for syslog.

Pre-Processing Syslog Sources

Attach a pre-processing Pipeline to most (or all) of your Syslog Sources. Use the pre-processing Pipeline to apply the same set of ingest-time processing that all syslog events will need, regardless of what subsequently happens on different subsets of the data.

Syslog data is a classic example of where a pre-processing Pipeline is useful: Unlike processing Pipelines, pre-processing Pipelines attach directly to a Source, enabling you to standardize or normalize what comes in. This way, you avoid having to implement that same functionality and logic separately in each processing Pipeline associated with a Syslog Source.

Pipeline for Selected Syslog Data Subsets

Configure dedicated Pipelines (and Routes!) for each distinct subset of data that arrives via syslog senders. These subsets might include DHCP logs from the router, traffic logs from the firewall, operational logs from load balancers, and virtualization logs from on-prem virtualization hypervisors.

Certain kinds of subset-specific processing have become Cribl Stream best practices. These include:

- For ESXi or other hypervisor logs, use the Drop Function to drop `debug`-level logs.
- For firewall logs, enrich with DNS names, GeoIP fields, and lookups from a threat list.
- For load balancer logs, use the Suppress Function to reduce volume.

Unless you're new to Cribl Stream, you've already created your own Pipelines, so we're not going to review that here. (If you are new to Cribl Stream, consider running through the free Cribl Stream Fundamentals sandbox course ASAP.)

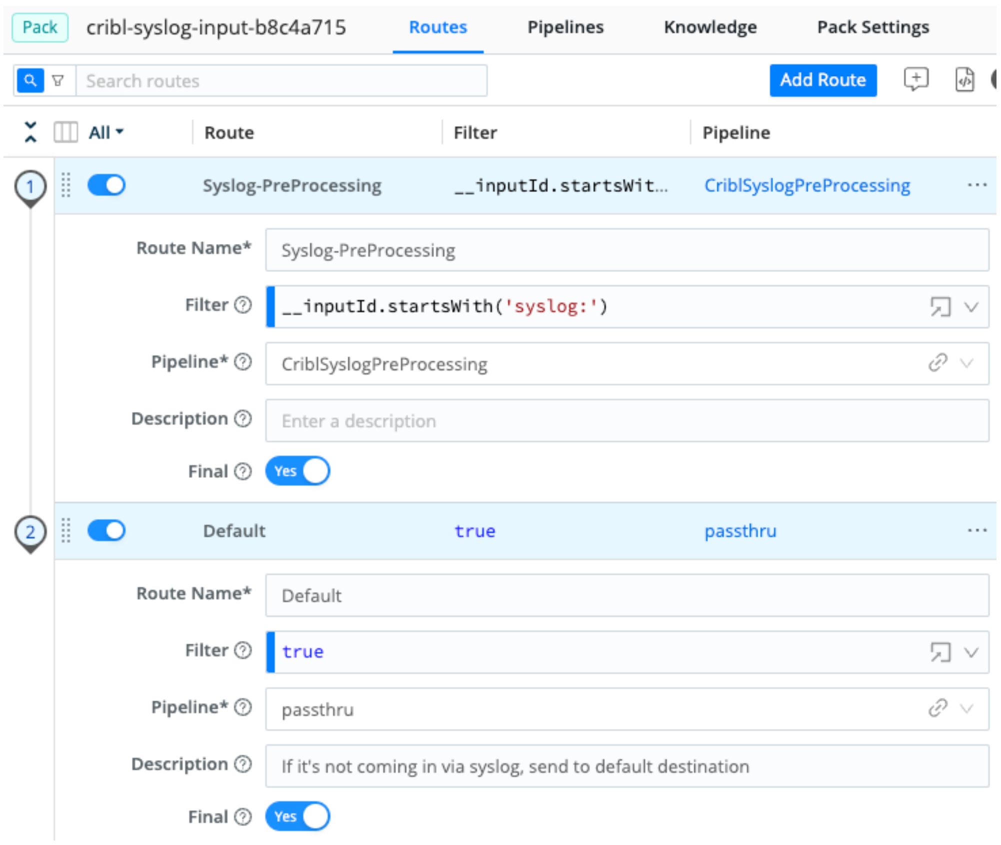
Importing the Pre-Processing Pipeline

Even before setting up a Cribl Stream Syslog Source, you'll want to install the Cribl Pack for Syslog Input (a.k.a. `cribl-syslog-input` Pack), which provides the required pre-processing Pipeline.

If this is your first time installing from the Cribl Dispensary, see the full directions with screenshots. Otherwise:

1. Open the Cribl Pack Dispenary's Cribl Pack for Syslog Input page.
2. Under **Releases** at right, click the link for the latest release.
3. In the resulting **Assets** section, right-click the `.crbl` file to download it locally.
4. In Cribl Stream, select a Worker Group, then select **Processing** > **Packs**.
5. Click the **Add New** button, and select **Import from file**.
6. Select the Pack you downloaded, and follow the prompts from there. In most cases, when installing a Dispensary Pack, you should not enter an override value for **New Pack ID**.
7. Review the Pack's `README` file, available in the Pack's **Settings** link and also online.

Let's examine what this Syslog Input Pack provides, starting with the Routes page.


Pack cribl-syslog-input-b8c4a715 | Routes | Pipelines | Knowledge | Pack Settings

Search routes | Add Route

All	Route	Filter	Pipeline
1	Syslog-PreProcessing	`__inputId.startsWit...`	CriblSyslogPreProcessing

Route Name*: Syslog-PreProcessing
Filter: `__inputId.startsWith('syslog:')`
Pipeline*: CriblSyslogPreProcessing
Description: Enter a description
Final: Yes

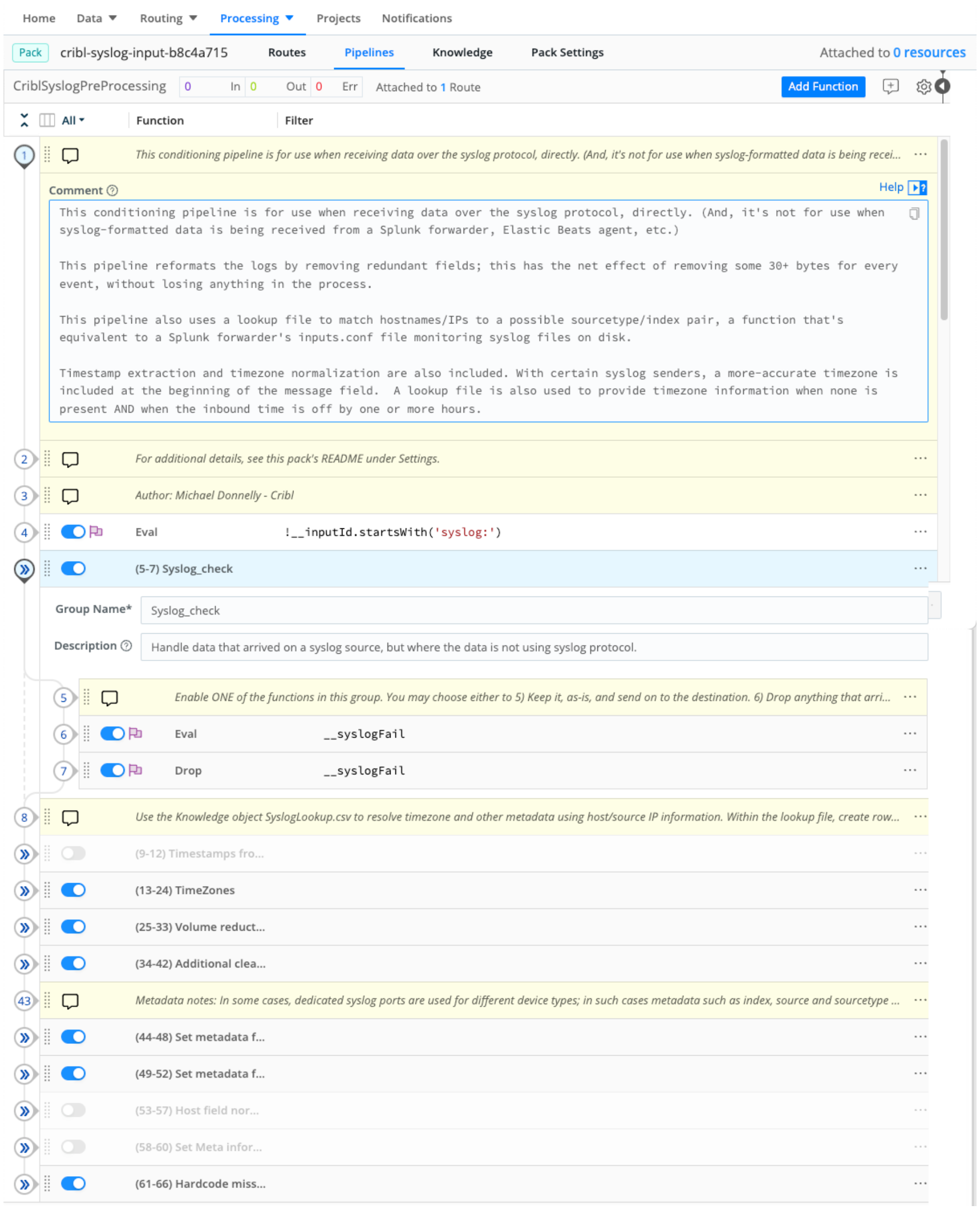
All	Route	Filter	Pipeline
2	Default	`true`	passthru

Route Name*: Default
Filter: `true`
Pipeline*: passthru
Description: If it's not coming in via syslog, send to default destination
Final: Yes

Routes page in the Cribl Pack for Syslog Input

The first Route matches based on `inputId`. Anything arriving via a Cribl Stream Syslog Source will match this filter, as long as you've configured the Source to use the Pack, as shown below.

On first installation of the Pack, the Pipeline defaults to commonly used configurations. Cribl **strongly** recommends reviewing the Pipeline's internal documentation to understand which features are enabled. This documentation consists of Comments for each section, and a Description for each Function to explain what's happening in that step.



1 *This conditioning pipeline is for use when receiving data over the syslog protocol, directly. (And, it's not for use when syslog-formatted data is being recei...*

Comment

```
This conditioning pipeline is for use when receiving data over the syslog protocol, directly. (And, it's not for use when
syslog-formatted data is being received from a Splunk forwarder, Elastic Beats agent, etc.)

This pipeline reformats the logs by removing redundant fields; this has the net effect of removing some 30+ bytes for every
event, without losing anything in the process.

This pipeline also uses a lookup file to match hostnames/IPs to a possible sourcetype/index pair, a function that's
equivalent to a Splunk forwarder's inputs.conf file monitoring syslog files on disk.

Timestamp extraction and timezone normalization are also included. With certain syslog senders, a more-accurate timezone is
included at the beginning of the message field.  A lookup file is also used to provide timezone information when none is
present AND when the inbound time is off by one or more hours.
```

2 *For additional details, see this pack's README under Settings.*

3 *Author: Michael Donnelly - Cribl*

4 Eval `!__inputId.startsWith('syslog:')`

(5-7) Syslog_check

Group Name*: Syslog_check

Description: Handle data that arrived on a syslog source, but where the data is not using syslog protocol.

5 *Enable ONE of the functions in this group. You may choose either to 5) Keep it, as-is, and send on to the destination. 6) Drop anything that arri...*

6 Eval `__syslogFail`

7 Drop `__syslogFail`

8 *Use the Knowledge object SyslogLookup.csv to resolve timezone and other metadata using host/source IP information. Within the lookup file, create row...*

(9-12) Timestamps fro...

(13-24) TimeZones

(25-33) Volume reduct...

(34-42) Additional clea...

43 *Metadata notes: In some cases, dedicated syslog ports are used for different device types; in such cases metadata such as index, source and sourcetype ...*

(44-48) Set metadata f...

(49-52) Set metadata f...

(53-57) Host field nor...

(58-60) Set Meta infor...

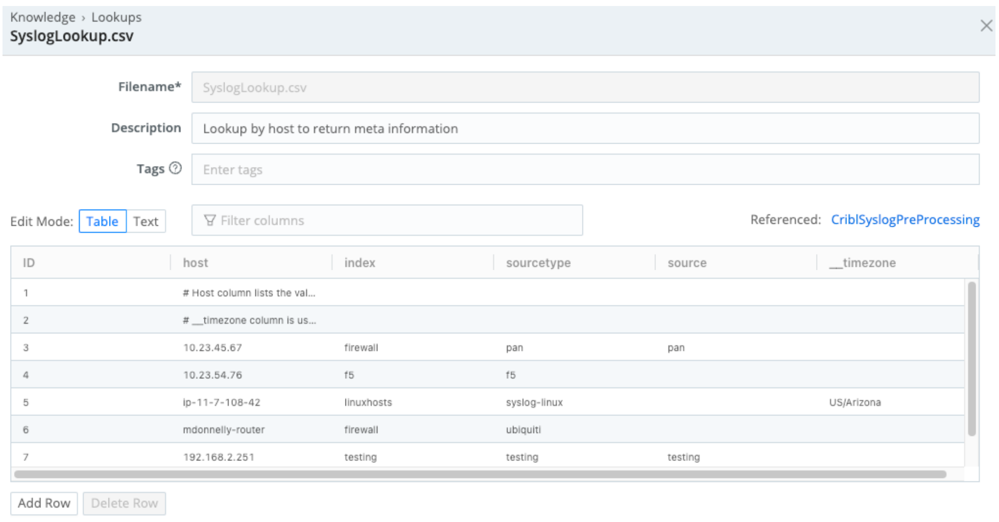
(61-66) Hardcode miss...

The Pipeline's internal documentation

Examining this documentation should make it clear what changes (if any) are needed to suit your deployment environment. Go ahead and make those changes now.

The Lookup File

The Pack ships with a lookup file called `SyslogLookup.csv`, whose contents you should replace as necessary. To access the file, navigate to **Processing** > **Packs**, select the `cribl-syslog-input` Pack, and click **Knowledge**.



Knowledge › Lookups
SyslogLookup.csv

Filename*: SyslogLookup.csv
Description: Lookup by host to return meta information
Tags: Enter tags

Edit Mode: Table Text — Filter columns — Referenced: CriblSyslogPreProcessing

ID	host	index	sourcetype	source	__timezone
1	# Host column lists the val...				
2	# __timezone column is us...				
3	10.23.45.67	firewall	pan	pan	
4	10.23.54.76	f5	f5		
5	ip-11-7-108-42	linuxhosts	syslog-linux		US/Arizona
6	mdonnelly-router	firewall	ubiquiti		
7	192.168.2.251	testing	testing	testing	

Stock `SyslogLookup.csv` file, to be filled with customer data

Creating Syslog Sources

Now that you've imported the pre-processing Pipeline, the next step is to ensure that you have the Syslog Sources you need.

Configure the `in_syslog` Source

Cribl Stream ships with a Syslog Source named `in_syslog`, which is 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.

Use the `in_syslog` Source for syslog senders that are hard-coded to send to port 514, as described above. In the **QuickConnect** UI: Click **Add Source**. From the resulting drawer's tiles, select [**Push** >] **Syslog**. Click **Select Existing**, then `in_syslog`.

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 [**Push** >] **Syslog**. In the **Manage Sources** page, click `in_syslog`.

Configure the fields and options as follows:

General Settings

- **Enabled**: Toggle to `Yes`.

- **UDP Port** and **TCP Port**: Assuming you're following the guidelines in this tutorial, you'll have a load balancer relaying incoming traffic from port 514 to port 9514. Therefore, leave the `in_syslog` Source configured to listen on its default port, 9514, for both TCP and UDP.

TLS Settings (TCP Only)

- Enabling TLS is strongly recommended if the Source will be receiving data across the Internet.

Processing Settings

- **Metadata**: There is no need to add fields here, because for generic senders coming in on 9514, the Pack will set the meta-information via the `SyslogLookup.csv` file. What you need to do is edit the lookup file to add content appropriate to your deployment.
- **Pre-processing**: From the **Pipeline** drop-down, select `PACK: cribl-syslog-input (Syslog Preprocessing)`.

Create a Source for Each Device Class

As explained above, you should now create a Syslog Source for each vendor/class of syslog sender. Create each Syslog Source as follows:

In the QuickConnect UI: Click **Add Source**. From the resulting drawer's tiles, select [**Push** >] **Syslog**. Click **Add New**.

If you use QuickConnect, remember that each Source/Destination pair will be parallel and independent.

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 [**Push** >] **Syslog**. Click **New Source** to open a **New Source** modal.

Configure the fields and options as follows:

General Settings

- **Input ID**: This field specifies the name for the Source, which will appear in the `__inputId` field as well as on Monitoring pages that show Source information. Common examples include `in_syslog_cisco_switch` for Cisco switches, `in_syslog_f5` for F5 load balancers, and so on.

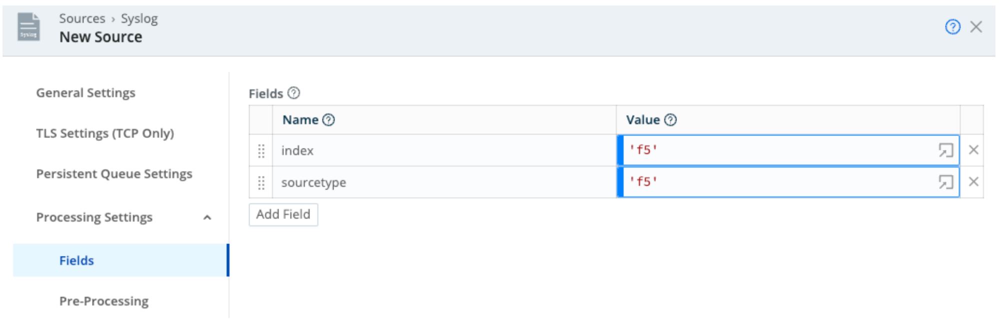
- **UDP Port** and **TCP Port**: Enter the dedicated port you have chosen for the device class, and use UDP or TCP according to the recommendations above.

TLS Settings (TCP Only)

- Enabling TLS is strongly recommended if the Source will be receiving data across the Internet.

Processing Settings

- **Metadata**: Select **Fields (Metadata)** and add the fields appropriate for the intended Destination(s), such as `sourcetype`, `index`, `__timezone`, and any other meta-information you want to tie to the sender's hostname or IP address.



Fields

Name	Value
index	'f5'
sourcetype	'f5'

Metadata fields tied to a dedicated-port Syslog source

- **Pre-processing**: From the **Pipeline** drop-down, select `PACK: cribl-syslog-input (Syslog Preprocessing)`.

Adding Processing Pipelines

Congratulations! You've gotten a nice grounding in syslog processing, and you've seen methods with which you can:

- Properly architect the deployment environment, where syslog sender data travels through a load balancer, which then distributes it across a Cribl Stream Worker Group.
- Import a Pack from the Cribl Pack Dispensary.
- Create new Cribl Stream Syslog Sources which use the **Cribl Pack for Syslog Input**.
- Hard-code meta-information for specific ports, or use the lookup file to map meta-information for specific hosts.

Your logical next step: using Pipelines and Routes, transform syslog data in a way that makes sense for your particular syslog sender(s).

We'll look at two use cases:

- QuickConnect a Syslog Sender to a Source
- Route Multiple Syslog Senders to a Source

The point is to see how Cribl Stream offers different approaches for different scenarios. For configuring one dedicated Source to receive a given, single dataset, QuickConnect is ideal. But to accommodate a mix of data arriving on the same Source and port, and needing to be divided into subsets, and processed differently by different Pipelines – that is where we need Routes.

Use Case A: QuickConnect a Syslog Sender to a Source

Of the many possible examples, we'll focus on reducing the volume of VMware ESXi syslog data by dropping events of `debug` severity.

Data reduction will be significant, because `debug` severity events can make up 80-90% of syslog data sent from ESXi servers. Not only that: Reducing the data volume reduces the CPU load and storage used to process the data, and searches will respond faster. In a world where we're searching for needles in a haystack, dropping 80% of the hay makes everything better.

To set this up:

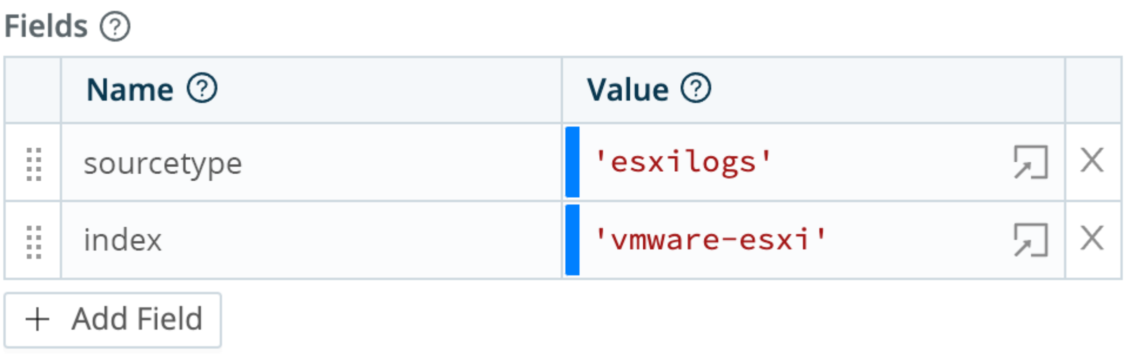
1. In your Worker Group, navigate to **Pipelines**, click **Add Pipeline**, then click **Add Pipeline**.
2. Name the new Pipeline `DropNoisySyslog` or something similar.
3. Click **Comment** and enter a good description of what you're doing. (This is an important best practice!)
4. Add a Drop Function with filter `severityName=='debug'`. (You could also use sampling, or dynamic sampling if you wanted to be fancy about this.)
5. Click **Save** and we're done.

Next, we need to direct VMware ESXi data to our Pipeline. We'll do this using QuickConnect, which is the fastest way to configure a new Source, tie it to a Destination, and assign pre-processing and processing Pipelines to that Source. (We could also do this with Routes; that simply requires a few more configuration steps.)

In the **QuickConnect** UI:

1. Click **Add Source**.
2. From the resulting drawer's tiles, select [**Push** >] **Syslog**.
3. Click **Add New**.
4. Configure the fields and options as follows:
   - **Name**: `in_syslog_ESXi`.

- **TCP**: `1517`. This is the TCP port on which ESXi servers send their logs.
- **Pre-Processing** > **Pack**: Select `cribl-syslog-input`.
- **Fields (metadata)**: Add a `sourcetype` field with value `esxilogs`, and an `index` field with value `vmware-esxi`.



Name	Value
sourcetype	'esxilogs'
index	'vmware-esxi'

Adding metadata appropriate for ESXi

5. Click **Save**.
6. On the main **QuickConnect** page, drag a connector from `in_syslog_ESXi` to each Destination you want to receive these events. When prompted, select **Pipeline**, select your `DropNoisySyslog`, and click **Save**.
7. Commit and deploy, and you're all set!

Use Case B: Route Multiple Syslog Senders to a Source

Let's imagine you have incoming data from a router. This data is tagged with `sourcetype=myrouter`, and it includes a mix of DHCP actions, login/authentication attempts, and firewall traffic logs.

Our goal is to send each of the three data subsets of data – DHCP, login, and firewall – to its own Pipeline.

We know that the **Cribl Pack for Syslog Input** has a Lookup Function that – for data arriving on the default port – should return meta-information such as `sourcetype` or `index`. We can combine this metadata with some matching of strings within the data, in Route filters that direct the right data to the right Pipeline.

> Although in reality, the `myrouter` firewall does not exist, the example that follows shows the art of the possible. We'll spell out the procedures as if the scenario were real. Perhaps one day you'll use it as a template for an actual deployment.

Create a Pipeline for Each Data Subset

- Create three Pipelines, naming each one for a data subset: `myrouter-dhcp`, `myrouter-auth`, and `myrouter-fw-traffic`.
- Leave the Pipelines empty for now; we'll add Functions later on.

Create a Route for Each Data Subset

1. Navigate to **Routing** > **Data Routes**.
2. Click **Add Route**.
3. Configure a Route as specified in the table below:

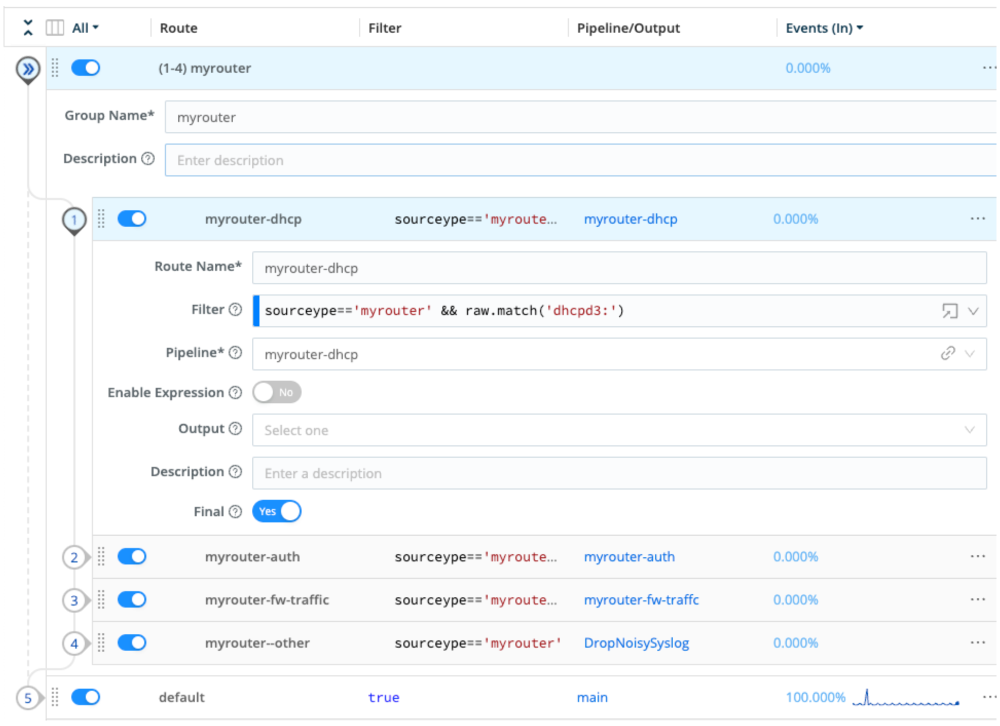
Name	Filter	Pipeline
`myrouter-dhcp`	`sourcetype=='myrouter' && raw.match('dhcpd3:')`	`myrouter-dhcp`
`myrouter-auth`	`sourcetype=='myrouter' && raw.match('login')`	`myrouter-auth`
`myrouter-fw-traffic`	`sourcetype=='myrouter' && raw.match('TRAFFIC')`	`myrouter-fw-traffic`
`myrouter-other`	`sourcetype=='myrouter'`	`DropNoisySyslog`

4. Repeat the preceding steps until you have configured four new Routes – one for each of the three data subsets, plus a final Route to drop events that don't match our Filters (i.e., noise).

All Routes should have the **Final** flag to `Yes`, and the output set to whatever Destination you think is appropriate for the given data subset.

5. Click the `myrouter-dhcp` Route's ••• (Options) menu, then select **Group Actions** > **Create Group**.
6. Name new Group `myrouter`.
7. Drag the Routes you created above into the new Group. (Be sure that `myrouter-other` is listed last.)
8. Drag the new Group to whichever vertical position makes the most sense for your deployment.

Once you've completed all the above steps, then after collapsing your new Routes, you should see something like this:



All	Route	Filter	Pipeline/Output	Events (In)
	(1-4) myrouter			0.000%
1	myrouter-dhcp	sourceype=='myroute...	myrouter-dhcp	0.000%
2	myrouter-auth	sourceype=='myroute...	myrouter-auth	0.000%
3	myrouter-fw-traffic	sourceype=='myroute...	myrouter-fw-traffc	0.000%
4	myrouter--other	sourceype=='myrouter'	DropNoisySyslog	0.000%
5	default	true	main	100.000%

Group Name*: myrouter
Description: Enter description

Route Name*: myrouter-dhcp
Filter: `sourceype=='myrouter' && raw.match('dhcpd3:')`
Pipeline*: myrouter-dhcp
Enable Expression: No
Output: Select one
Description: Enter a description
Final: Yes

Example of Routes in a Group, each tied to the same sender

Next, edit each of the `myrouter-<data-subset>` pipelines:

- Use an Eval Function to modify the `sourcetype` (and perhaps other metadata.)
- Use appropriate Functions to accomplish the enrichment, suppression, volume reduction, and/or other transformations that the data needs.

Takeaways from This Routing / Pipeline Use Case

- Routing provides the flexibility needed when dealing with multiple datasets from a single sender, or when a single Source receives multiple datasets.
- Ensure that you have a dedicated Pipeline (and Route) for each discrete dataset that you want to manipulate in some way (e.g., `fw-traffic`).
- Ensure that you have a final Route that matches the general data from the overall sourcetype.
- Use the Routing page's Groups option to create logical containers for sets of Routes that belong together.

17.6.4.2. Palo Alto Syslog To Cribl.Cloud

Panorama management appliances give you insights into what is happening behind your Palo Alto Networks firewalls, including details about applications, URLs, threats, data files, and patterns. By forwarding logging data directly from Palo Alto firewalls or Panorama management appliances to your Cribl.Cloud instance, you can shape, process, search, and route the data to the analytics platform of your choice.

In this example, we will show you how to configure your devices so that you can connect to your Cribl.Cloud instance.

Obtain Cribl.Cloud CA Certificates

The first step is to obtain the CA chain from your Cribl.Cloud Organization. This script automates the process.

The `HOST` domain in this example uses the `default` Worker Group. Modify this substring, as needed, to query other Cribl.Cloud-managed Worker Groups.

```
mkdir certs && cd certs
HOST=default.main.<OrganizationID>.cribl.cloud
PORT=6514
echo | openssl s_client -servername $HOST -showcerts -connect $HOST:$PORT 2> /dev/r

for cert in $(ls -1); do
  newname=$(openssl x509 -noout -subject -in $cert | sed -nE 's/.*CN ?= ?(.*)/\1/;
  echo "${newname}"; mv "${cert}" "${newname}"
done
```

Example output:

```
thegoat@in certs % ls -la
total 24
drwxr-xr-x    5 thegoat  staff   160 Sep 13 21:45 .
drwx------@ 44 thegoat  staff  1408 Sep 13 20:45 ..
-rw-r--r--    1 thegoat  staff  2472 Sep 13 21:44 logstream_<OrganizationID>_cribl_
-rw-r--r--    1 thegoat  staff  1968 Sep 13 21:44 usertrust_rsa_certification_autho
-rw-r--r--    1 thegoat  staff  2431 Sep 13 21:44 zerossl_rsa_domain_secure_site_ca
```

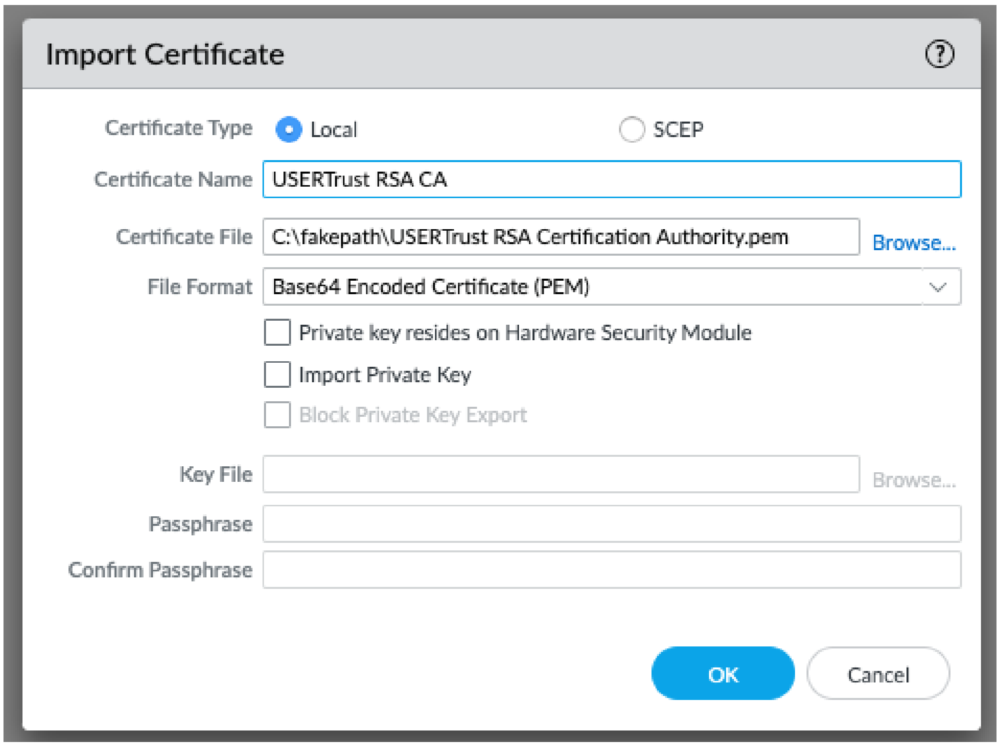
Import the `usertrust` and `zerossl` certs in the next section.

Import Trusted Roots into PAN

Next, you need to import the certificates into PAN. In the PAN OS:

- Go to **Device** > **Certificate Management** > **Certificates**.
- Click the **Import** button at the bottom.
- The **Import Certificate** configuration modal (below) appears. Import the two certificates obtained in the previous section.

We configured the steps below on PAN OS `10.2.2`. The screenshots might vary depending on your software version, and on whether you are using PAN OS vs. Panorama.

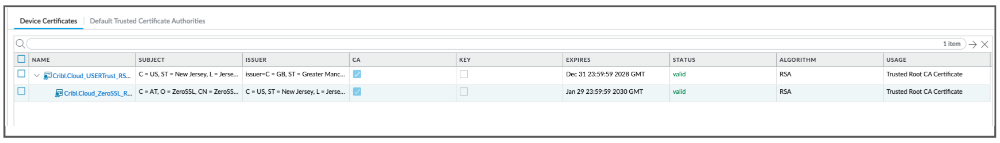


Import Certificate

Certificate Type	[✓ Local] SCEP
Certificate Name	USERTrust RSA CA
Certificate File	C:\fakepath\USERTrust RSA Certification Authority.pem
File Format	Base64 Encoded Certificate (PEM)
	☐ Private key resides on Hardware Security Module
	☐ Import Private Key
	☐ Block Private Key Export
Key File	
Passphrase	
Confirm Passphrase	

Import Certificate

- Once you have imported both certificates, go to the **Device Certificates** table and click on the blue link for each certificate you just imported.



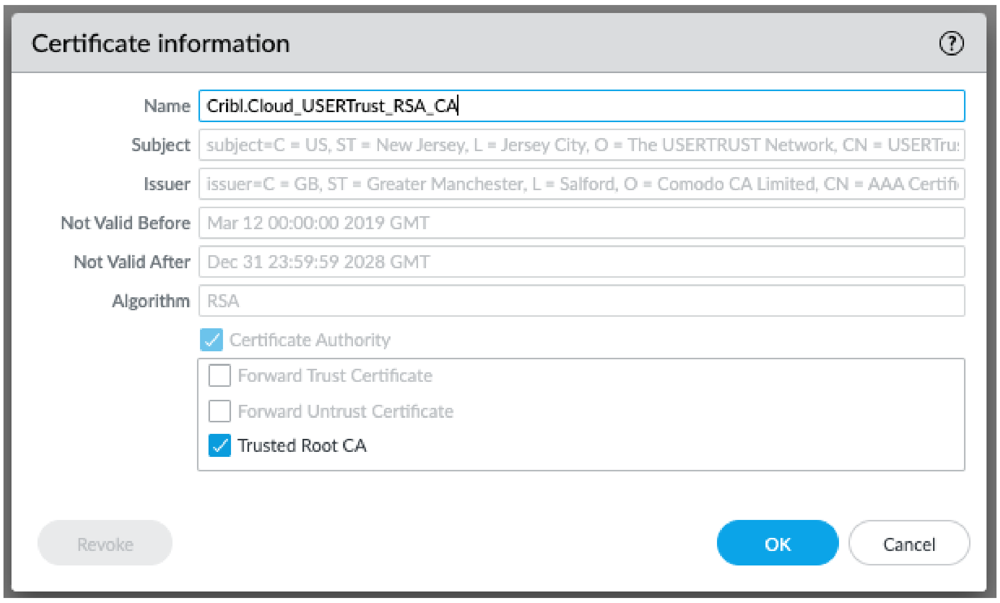
Device Certificates | Default Trusted Certificate Authorities

1 item

NAME	SUBJECT	ISSUER	CA	KEY	EXPIRES	STATUS	ALGORITHM	USAGE
Cribl.Cloud_USERTrust_RS...	C = US, ST = New Jersey, L = Jerse...	issuer=C = GB, ST = Greater Manc...	✓	☐	Dec 31 23:59:59 2028 GMT	valid	RSA	Trusted Root CA Certificate
Cribl.Cloud_ZeroSSL_R...	C = AT, O = ZeroSSL, CN = ZeroSS...	C = US, ST = New Jersey, L = Jerse...	✓	☐	Jan 29 23:59:59 2030 GMT	valid	RSA	Trusted Root CA Certificate

Device Certificates

- Select the **Trusted Root CA** check box for each certificate.


Certificate information

Name	Cribl.Cloud_USERTrust_RSA_CA
Subject	subject=C = US, ST = New Jersey, L = Jersey City, O = The USERTRUST Network, CN = USERTru:
Issuer	issuer=C = GB, ST = Greater Manchester, L = Salford, O = Comodo CA Limited, CN = AAA Certifi
Not Valid Before	Mar 12 00:00:00 2019 GMT
Not Valid After	Dec 31 23:59:59 2028 GMT
Algorithm	RSA

✓ Certificate Authority

☐ Forward Trust Certificate

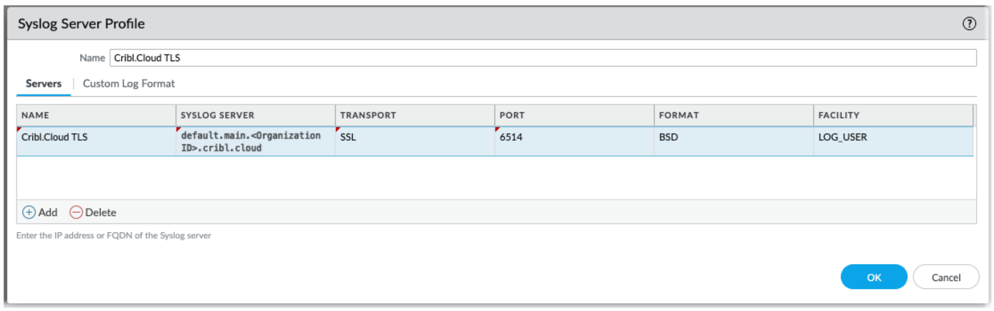
☐ Forward Untrust Certificate

✓ Trusted Root CA

Certificate information

Configure the Syslog Server Profile

After importing the certs, create a new Syslog Server Profile in **Device** > **Server Profiles** > **Syslog**.


Syslog Server Profile

Name Cribl.Cloud TLS

NAME	SYSLOG SERVER	TRANSPORT	PORT	FORMAT	FACILITY
Cribl.Cloud TLS	default.main.<Organization ID>.cribl.cloud	SSL	6514	BSD	LOG_USER

Enter the IP address or FQDN of the Syslog server

Syslog Server Profile

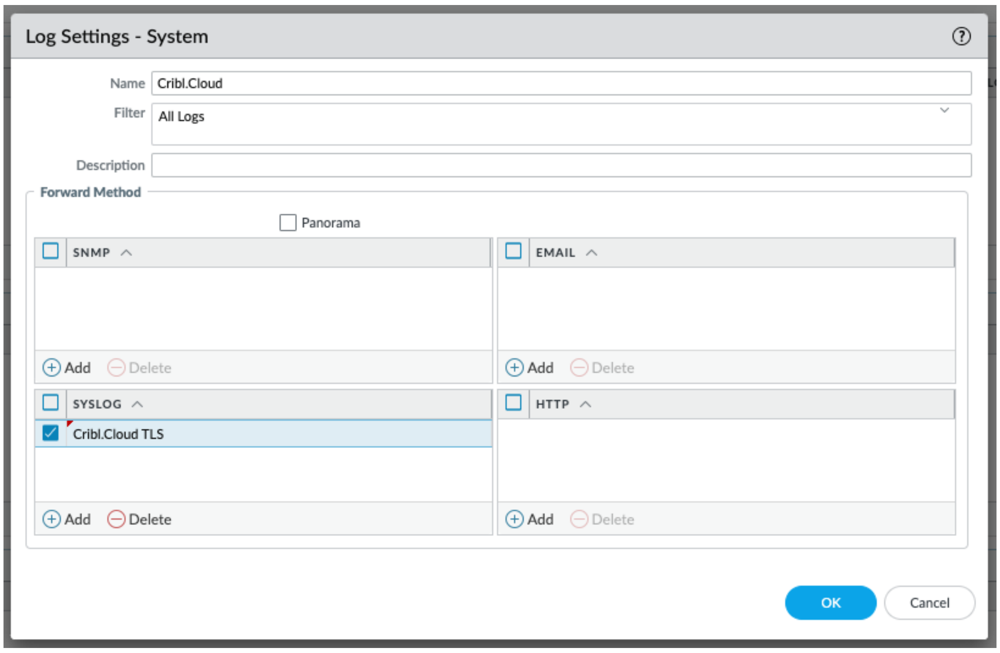
The Syslog server is the ingest address to your Cribl.Cloud Organization. To find your ingest address, click **Network Settings** at the top of your Organization's portal and select the Worker Group you're sending to. For example, sending to the `default` Group your ingest address would look like:

```
default.main.<OrganizationID>.cribl.cloud
```

Configure Log Settings to Use the Syslog Profile

Next, configure your log settings to use the Syslog Profile.

Go to **Device** > **Log Settings** and configure each relevant section with the appropriate settings.


Log Settings - System

Name: Cribl.Cloud
Filter: All Logs
Description:
Forward Method
Panorama

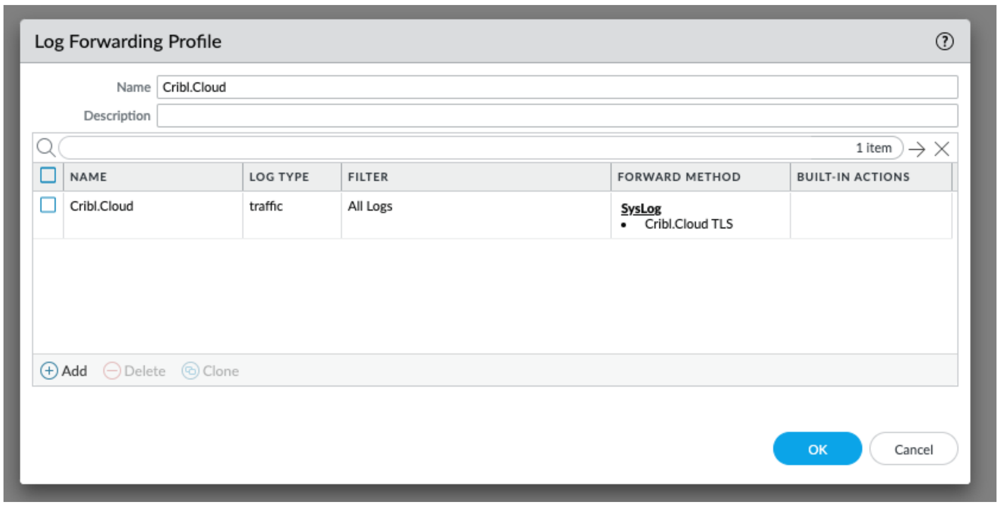
SYSLOG
✓ Cribl.Cloud TLS

Log Settings

Configure Firewall Rule Log Forwarding

Use the **Log Forwarding Profile** to create a ruleset and configure the appropriate log types to be forwarded.

Go to **Objects** > **Log Forwarding** and configure each relevant section with the appropriate settings.


Log Forwarding Profile

Name: Cribl.Cloud
Description:

NAME	LOG TYPE	FILTER	FORWARD METHOD	BUILT-IN ACTIONS
Cribl.Cloud	traffic	All Logs	SysLog • Cribl.Cloud TLS	

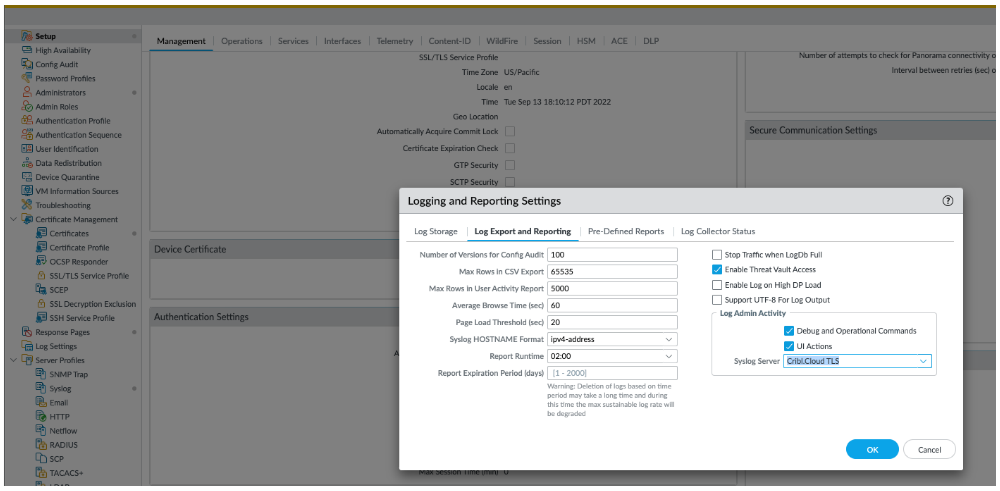
Log Forwarding Profile

Forward Admin Activity Logs

Admin activity logs can provide more context to your logs. To forward your admin activity logs:

Go to **Device** > **Setup** > **Management**, and click the gear icon in the **Logging and Reporting** settings.

Enable the **Debug and Operational Commands** and **UI Actions**, and choose the appropriate Syslog server.



Logging and Reporting Settings

Log Storage | **Log Export and Reporting** | Pre-Defined Reports | Log Collector Status

Number of Versions for Config Audit	100
Max Rows in CSV Export	65535
Max Rows in User Activity Report	5000
Average Browse Time (sec)	60
Page Load Threshold (sec)	20
Syslog HOSTNAME Format	ipv4-address
Report Runtime	02:00
Report Expiration Period (days)	[1 - 2000]

Warning: Deletion of logs based on time period may take a long time and during this time the max sustainable log rate will be degraded

- ☐ Stop Traffic when LogDb Full
- ✓ Enable Threat Vault Access
- ☐ Enable Log on High DP Load
- ☐ Support UTF-8 For Log Output

Log Admin Activity
- ✓ Debug and Operational Commands
- ✓ UI Actions
- Syslog Server: Cribl.Cloud TLS

Forwarding admin activity logs

Customize OCSP Validation Settings for the `syslog-ng` Process

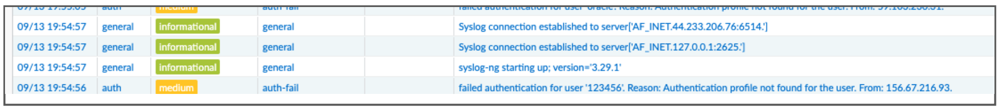
If the certificate validation fails, follow these steps to customize the validation settings for the `syslog-ng` process. You'll need to shell into the box to perform them.

These commands are for an individual PAN OS device, and might be different for Panorama:

```
set syslogng ssl-conn-validation all-conns skip
set syslogng ssl-conn-validation explicit OCSP skip CRL skip EKU skip
```

Validation

To validate whether you have successfully set up the TLS-encrypted Syslog, check the logs on your PAN OS.



```
09/13 19:54:57  general  informational  general   Syslog connection established to server['AF_INET.44.233.206.76:6514.']
09/13 19:54:57  general  informational  general   Syslog connection established to server['AF_INET.127.0.0.1:2625.']
09/13 19:54:57  general  informational  general   syslog-ng starting up; version='3.29.1'
09/13 19:54:56  auth     medium         auth-fail failed authentication for user '123456'. Reason: Authentication profile not found for the user. From: 156.67.216.93.
```

Validating syslog setup

Troubleshooting

If you run into issues not anticipated above, you can use some of the following commands to further diagnose them:

```
# restart the syslog-ng process
debug syslog-ng restart

# see syslog-ng process logs
tail follow yes mp-log syslog-ng.log

# get stats for processes
debug log-receiver statistics
debug syslog-ng stats

# tcpdump activity on the syslog port and view the results
# note: control-c to exit the tcpdump, it will not show anything on the cli when ru
tcpdump filter "port 6514" snaplen 0
view-pcap mgmt-pcap mgmt.pcap
```

17.6.5. Configuring Upstream Logging Agents

This page explains how to quickly connect a wide selection of common logging agents and other log sources to Cribl Stream. The examples below are equally valid for Cribl.Cloud and customer-managed Cribl Stream instances.

Whichever source you choose, start by doing a live capture on the corresponding Source in Cribl Stream. Verify that data is coming in. Then save your capture to a sample file, which will help you build your Pipelines.

Fluent Bit and Fluentd

Both Fluent Bit (written in C) and fluentd (written in Ruby) are open-source log collectors, processors, and aggregators.

Fluent Bit to HEC

Of Fluent Bit's many output options, several work with Cribl Stream. Here's how to connect a Cribl Stream Splunk HEC Source to Fluent Bit with the Splunk HEC formatting option.

You'll need to copy and paste the following:

- From the Cribl.Cloud **Network Settings** > **Data Sources** tab, the `in_splunk_hec` **Ingest Address**.
- From Cribl Stream's **Sources** > **Splunk HEC** > `<source_name>` > **Auth Tokens** tab, the HEC token.

Decide how to adjust the `match` parameter for your use case. For example, an asterisk (wildcard) value will send **all** events to Cribl Stream.

Edit the Fluent Bit settings accordingly. They're usually in `/etc/td-agent-bit/td-agent-bit.conf`, in a form similar to this:

```
[OUTPUT]
    name         splunk
    match        *
    host         in.main-default-<organization>.cribl.cloud
    port         8088
    splunk_token <HEC_token>
    tls          on
    # optional
```

Save the changes and restart the `td-agent-bit` service.

Fluent Bit to Elastic Sources

You can also connect Cribl Stream to Fluent Bit, using an Elastic Source.

```
[OUTPUT]
    name          es
    match         *
    host          10.0.21.134
    port          9200
    index         my_index
    type          my_type
```

Fluentd to HEC

If you don't already have it, install the `splunk_hec` fluentd mod:

```
sudo gem install fluent-plugin-splunk-hec
```

You'll need to copy and paste the following:

- From the Cribl.Cloud **Network Settings** > **Data Sources** tab, the `in_splunk_hec` **Ingest Address**.
- From Cribl Stream's **Sources** > **Splunk HEC** > <source_name> > **Auth Tokens** tab, the HEC token.

Decide how to adjust the `match` parameter for your use case. Use an asterisk (wildcard) if you want to send **all** events to Cribl Stream.

Edit the `<match` section of the fluentd settings accordingly. They could be in `/etc/fluent/fluent.conf` or another location, as described in the fluentd docs.

```
<match **>
  @type splunk_hec
  @log_level info
  hec_host in.main-default-<organization>.cribl.cloud
  hec_port 8088
  hec_token <HEC_token>
  index <index_name>
  source_key <file_path>
  <format>
    @type json
  </format>
</match>
```

Save the changes and restart fluentd.

Splunk

Cribl Stream can directly ingest the native Splunk2Splunk (S2S) protocol. As a result, both Splunk universal and heavy forwarders can send log data to Cribl Stream.

Splunk Forwarder (Universal or Heavy) to Splunk TCP

In Cribl Stream, create a Splunk TCP Source.

Cribl recommends setting an authorization token so that your receiver will accept only your traffic. Define this setting on your Splunk TCP Source config's **Auth Tokens** tab.

If you're using a Splunk heavy forwarder, you'll also want to decide whether to create filters in Splunk, or to let all of your traffic through to Cribl Stream. In the snippet below, we assume that you want all traffic delivered.

Adapt the following snippet to your use case and add it to `$splunk/etc/system/local/outputs.conf`.

```
[tcpout]
disabled        = false
defaultGroup    = criblcloud

[tcpout:criblcloud]
server          = in.main-default-<organization>.cribl.cloud:9997
sslRootCAPath   = $SPLUNK_HOME/etc/auth/cacert.pem
useSSL          = true
sendCookedData = true
token           = <optional-but-recommended>
```

Save the changes and restart the Splunk service.

Elastic

In the Elastic ecosystem, Elastic Filebeat is an agent that functions as a lightweight shipper for forwarding and centralizing log data.

Elastic Filebeat

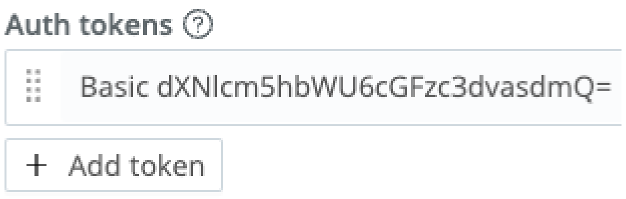
Cribl Stream can ingest the native Elasticsearch streaming protocol directly.

Cribl recommends setting an authorization token so that your receiver will accept only your traffic. This setting is in the **Auth Tokens** field of Cribl Stream's **Sources** > **Elasticsearch API** > **General Settings** tab.

If you want to enable Basic authentication:

- Concatenate your username and password with a colon in between, like this: `username:password`.
- Encode the `username:password` string using base64.
- Prepend the string `Basic` (including the trailing space) to the encoded string.
- Enter the resulting string in the **Auth tokens** field as described above.

You'll see something like this:



Basic dXNlcm5hbWU6cGFzc3dvasdmQ=

Basic auth in the Cribl Stream Elasticsearch API Source

```
output.elasticsearch:
  hosts: ["in.main-default-<organization>.cribl.cloud:9200/search"]
  protocol: "https"
  ssl.verification_mode: "full"
  username: <some_username>
  password: <some_password>
```

syslog

To receive syslog data in Cribl Stream, create a native Syslog Source. See Cribl's best practices for working with syslog data.

Cribl recommends not using UDP over the public internet. UDP examples are included here for test purposes only.

UDP, by definition, does not guarantee delivery. While TCP mitigates this problem, using either UDP or TCP without TLS over the public internet exposes unencrypted data.

syslog-ng (TLS over TCP)

Cribl recommends using TLS over TCP to send data from syslog-ng to Cribl Stream.

Here's an example of `syslog-ng.conf` edited such that all event data will be encrypted on the way to Cribl.Cloud. Adapt the snippet to your use case and edit `syslog-ng.conf` accordingly.

```
destination d_syslog {
    syslog("in.main-default-<organization>.cribl.cloud"
        transport("tls")
        port(6514)
        tls(
            peer-verify(required-trusted)
            ca-dir("/etc/syslog-ng/ca.d")
        )
    );
};

log {
    source(s_network);
    destination(d_syslog);
};
```

Save the changes and restart the syslog-ng service.

syslog-ng (UDP or TCP)

Here's an example of `syslog-ng.conf` – for test purposes only, because it does not use TLS.

Adapt the snippet to your use case, and edit `syslog-ng.conf` accordingly. You can change `tcp` to `udp` to switch protocols.

```
destination d_syslog {
    syslog("in.main-default-<organization>.cribl.cloud"
        transport("tcp")
        port(9514)
    );
};

log {
    source(s_network);
    destination(d_syslog);
};
```

Save the changes and restart the syslog-ng service.

syslog-ng with Load-Balanced Outputs

You can forward logs from syslog-ng to multiple Cribl Stream Workers through a redundant, load-balanced output by using the syslog-ng `elasticsearch-http` output module. With this arrangement, syslog-ng receives syslog messages over TCP and/or UDP and outputs events over the Elasticsearch HTTP Bulk API to Cribl Stream Workers.

> Forwarding to Cribl Stream from the syslog-ng `elasticsearch-http` output module requires syslog-ng version 3.19 or higher.
>
> To check your syslog-ng version, run the following command:
>
> ```
> syslog-ng -V
> ```

Adapt the snippet to your use case, and edit `syslog-ng.conf` accordingly. To learn more about syslog-ng destination flags, see the syslog-ng Administration Guide.

```
@version: 3.33
@include "scl.conf"

# Inputs
source s_network {
    # flags(no-parse) disables syslog-ng parsing of syslog messages (raw passthru)
    network(transport("tcp") port("9514") flags(no-parse));
    network(transport("udp") port("9514") flags(no-parse));
};

destination d_elasticsearch_tls {
    # https://www.syslog-ng.com/technical-documents/doc/syslog-ng-open-source-edit
    elasticsearch-http(url("https://worker1:9200/_bulk" "https://worker2:9200/_bul
        batch-lines(100)
        batch-bytes(512Kb)
        batch-timeout(2500)
        persist-name("d_elasticsearch-http-load-balance")
        type("")
        index("syslog")
        # Auth token
        #headers("Authorization: <token from Stream>")
        # or Basic Auth
        #user("username")
        #password("password")
    );
};

log {
    source(s_network);
    destination(d_elasticsearch_tls);
};
```

Save the changes and restart the syslog-ng service.

rsyslog (TLS over TCP)

You'll need to install the `rsyslog-gnutls` package if it's not already on your system. E.g.:

```
apt install rsyslog-gnutls
```

If the `rsyslog-gnutls package` is not already on your system, run the following command (or equivalent) to install it:

```
apt install rsyslog-gnutls
```

Cribl recommends using TLS over TCP to send data from rsyslog to Cribl Stream.

Here's an example of `rsyslog.conf`, edited such that all event data will be encrypted on the way to Cribl.Cloud. Adapt the snippet to your use case, and edit `rsyslog.conf` accordingly:

```
# path to the crt file.
# You will need a valid ca cert file. Most linux distros come with this
$DefaultNetstreamDriverCAFile /etc/ssl/certs/ca-certificates.crt

*.*     action(
         type="omfwd"
         target="in.main-default-<organization>.cribl.cloud"
         port="6514"
         protocol="tcp"
         action.resumeRetryCount="100"
         queue.type="linkedList"
         queue.size="10000"
         StreamDriver="gtls"
         StreamDriverMode="1" # run driver in TLS-only mode
)
```

Save the changes and restart the rsyslog service.

rsyslog (UDP or TCP)

Here's an example of `rsyslog.conf` – for test purposes only, because it does not use TLS.

Adapt the snippet to your use case, and edit `rsyslog.conf` accordingly. You can change `tcp` to `udp` to switch protocols.

```
*.*     action(
         type="omfwd"
         target="in.main-default-<organization>.cribl.cloud"
         port="9514"
         protocol="tcp"
         action.resumeRetryCount="100"
         queue.type="linkedList"
         queue.size="10000"
)
```

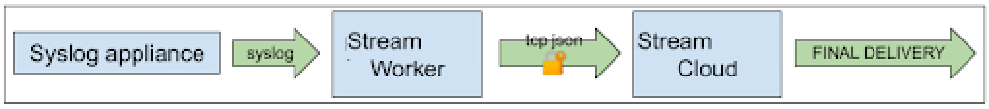
Save the changes and restart the rsyslog service.

Appliances that Send syslog

Many appliances emit syslog data. At a minimum, configure your appliance(s) to send syslog over TCP with TLS enabled, if possible.

Even better, stand up a Cribl Stream instance close to the appliances in your network topology. Have the appliances send syslog to the nearby Cribl Stream, which can optionally transform, filter, and/or enhance the data, and then send it to a more centralized Cribl Stream cluster in Cribl.Cloud or a customer-managed location.

The Cribl Stream worker that's closest to the appliance will use TCP JSON, with TLS enabled, to send syslog to the second Cribl Stream instance:



Sending syslog from an appliance to Cribl Cloud

The benefits of placing the first Cribl Stream instance close to the log producer include:

- Since there are fewer network hops, UDP becomes a more viable option, if you want it.
- If you're using TCP, that can be more performant here than it would be across the internet.
- The first Cribl Stream instance can do data reduction and/or compression. These operations reduce the volume of the data hitting the wire on the way to the next hop.
- This can help keep data egress costs under control.

Vector

Vector is Datadog's tool for building observability pipelines.

Vector to HEC

Vector supports Splunk's HTTP Event Collector (HEC). Here's how to connect a Cribl Stream Splunk HEC Source to Vector.

For Cribl.Cloud especially, Cribl recommends setting an authorization token so that your receiver will accept only your traffic. This setting is in Cribl Stream's **Sources** > **Splunk HEC** > **Auth Tokens** tab.

The snippet below references a random syslog generator called `dummy_logs` for testing purposes. Adapt the snippet to your use case and edit `vector.toml` (or `vector.yaml` or `vector.json`) accordingly.

```
[sinks.my_sink_id]
type           = "splunk_hec"
inputs         = [ "dummy_logs" ]
endpoint       = "https://in.main-default-<organization>.cribl.cloud:8088"
compression    = "gzip"
token          = "<optional-but-recommended>"
encoding.codec = "json"
```

Vector to Elasticsearch

Vector supports Elasticsearch API. Here's how to connect a Cribl Stream Elasticsearch API Source to Vector.

The snippet below references a random syslog generator called `dummy_logs` for testing purposes. The snippet also specifies Basic authentication. Follow the procedure for enabling Basic authentication described in the Elastic Filebeat example above.

Adapt the snippet to your use case and edit `vector.toml` (or `vector.yaml` or `vector.json`) accordingly.

```
[sinks.elastic_test]
type          = "elasticsearch"
inputs        = [ "dummy_logs" ]
endpoint      = "https://in.main-default-<organization>.cribl.cloud:9200/search"
compression   = "gzip"
index         = "my_es_index"
auth.user     = "<some_username>"
auth.password = "<some_password>"
auth.strategy = "basic"
```

Pull-based Sources

All the above data sources **push** data into Cribl Stream. Here are brief notes on configuring some popular Cribl Stream-native Pull Sources:

Amazon Kinesis

You can create an Amazon Kinesis Source in Cribl Stream. This is straightforward for customer-managed Cribl Stream instances.

Cribl.Cloud sits behind an network load balancer, which Kinesis Firehose does not support. For this reason, the best option for connecting Amazon Kinesis to Cribl.Cloud is to (1) have Kinesis Firehose write to an Amazon S3 bucket, and (2) set up a Cribl Stream Amazon S3 Source to pull data from there, as described in the next section.

Amazon S3

Configuring an S3 Source is virtually the same for Cribl.Cloud as it is for customer-managed Cribl Stream instances.

See this in-depth article about the Cribl Stream S3 Source, whose content also applies to Cribl.Cloud. Here's an overview of the process:

- Configure an S3 bucket to send `s3:ObjectCreated:*` events to an SQS queue.
- Configure a Cribl Stream S3 (Pull) Source (not Collector) to subscribe to the SQS feed from step 1.
  - Be sure to properly configure permissions for the Secret/Access keys you use for authentication.
- Set up Event Breaker rules as required based on the contents of your log files.

Office 365 Service, Activity, or Message Trace

Configuring an Office 365 Services, Activity, or Message Trace Source is virtually the same for Cribl.Cloud as it is for customer-managed Cribl Stream instances. Note that you need to start your Office 365 Content subscription from within Office 365, or there will be no data available to pull.

AppScope

Integrating AppScope with Cribl Stream is simple and fast. The easiest way is to just set the `$SCOPE_CRIBL` environment variable to define a connection between Cribl Stream and AppScope.

For example, you could "scope" the `nginx` command, and send the captured data using TLS over TCP:

```
SCOPE_CRIBL=tcp://in.main-default-<organization>.cribl.cloud:10091 scope nginx
```

17.6.6. BigPanda/Webhook Integration

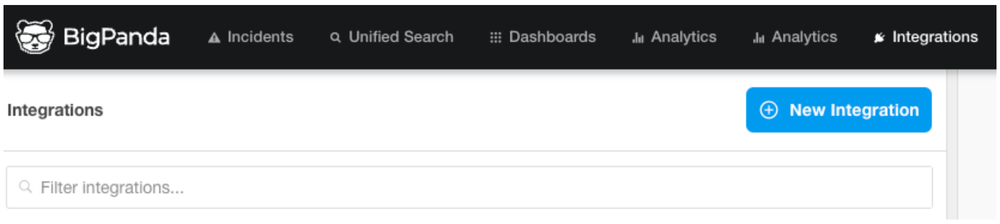
You can configure Cribl Stream to send Webhook notifications to the BigPanda IT Ops platform. These notifications arrive in BigPanda as Alerts, which BigPanda correlates into Incidents.

Before you begin, you should have an Admin account on a BigPanda Cloud instance.

Prepare BigPanda to Receive Data from Cribl Stream

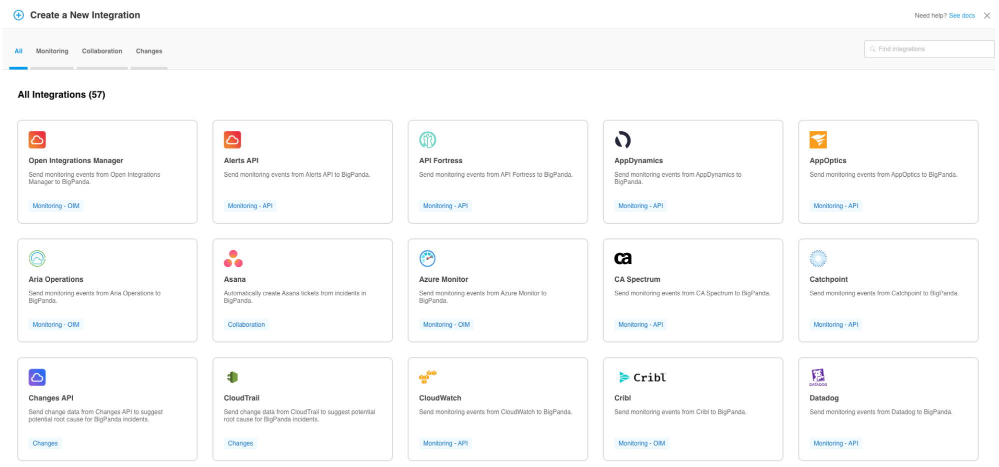
The BigPanda **App Key** and **Access Token** are separate and independent. The **Access Token** is a 32-character string that is part of the value that BigPanda generates for the `Authorization` HTTP header. (It functions like an auth token or bearer token.)

1. Log into your BigPanda Cloud instance as an Admin.
2. In the **Integrations** tab, click **New Integration**.



The **Integrations** tab

3. In the **Create a New Integration** modal, select the Cribl tile.



All Integrations (57)

Open Integrations Manager
Send monitoring events from Open Integrations Manager to BigPanda.
Monitoring - OIM

Alerts API
Send monitoring events from Alerts API to BigPanda.
Monitoring - API

API Fortress
Send monitoring events from API Fortress to BigPanda.
Monitoring - API

AppDynamics
Send monitoring events from AppDynamics to BigPanda.
Monitoring - API

AppOptics
Send monitoring events from AppOptics to BigPanda.
Monitoring - API

Aria Operations
Send monitoring events from Aria Operations to BigPanda.
Monitoring - OIM

Asana
Automatically create Asana tickets from incidents in BigPanda.
Collaboration

Azure Monitor
Send monitoring events from Azure Monitor to BigPanda.
Monitoring - OIM

CA Spectrum
Send monitoring events from CA Spectrum to BigPanda.
Monitoring - API

Catchpoint
Send monitoring events from Catchpoint to BigPanda.
Monitoring - API

Changes API
Send change data from Changes API to suggest potential root cause for BigPanda incidents.
Changes

CloudTrail
Send change data from CloudTrail to suggest potential root cause for BigPanda incidents.
Changes

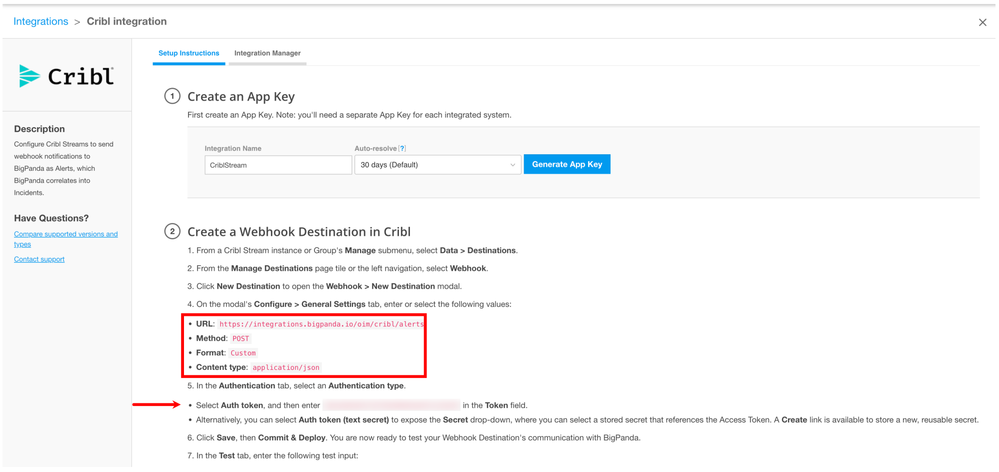
CloudWatch
Send monitoring events from CloudWatch to BigPanda.
Monitoring - API

Cribl
Send monitoring events from Cribl to BigPanda.
Monitoring - OIM

Datadog
Send monitoring events from Datadog to BigPanda.
Monitoring - API

The **Create a New Integration** modal

- This opens the **Cribl Integration** page.


Integrations > Cribl integration

Setup Instructions | Integration Manager

Description

Configure Cribl Streams to send webhook notifications to BigPanda as Alerts, which BigPanda correlates into Incidents.

Have Questions?

Compare supported versions and types

Contact support

① Create an App Key

First create an App Key. Note: you'll need a separate App Key for each integrated system.

Integration Name: CriblStream

Auto-resolve [?]: 30 days (Default)

Generate App Key

② Create a Webhook Destination in Cribl

1. From a Cribl Stream instance or Group's **Manage** submenu, select **Data > Destinations**.
2. From the **Manage Destinations** page tile or the left navigation, select **Webhook**.
3. Click **New Destination** to open the **Webhook > New Destination** modal.
4. On the modal's **Configure > General Settings** tab, enter or select the following values:
   - **URL**: `https://integrations.bigpanda.io/oim/cribl/alerts`
   - **Method**: `POST`
   - **Format**: `Custom`
   - **Content type**: `application/json`
5. In the **Authentication** tab, select an **Authentication type**.
   - Select **Auth token**, and then enter [redacted] in the **Token** field.
   - Alternatively, you can select **Auth token (text secret)** to expose the **Secret** drop-down, where you can select a stored secret that references the Access Token. A **Create** link is available to store a new, reusable secret.
6. Click **Save**, then **Commit & Deploy**. You are now ready to test your Webhook Destination's communication with BigPanda.
7. In the **Test** tab, enter the following test input:

The **Cribl Integration** page

4. In the **Create an App Key** section, generate an App Key named `Cribl Stream`. You'll need the App Key when configuring Cribl Stream in the next section. Cribl Stream will insert the App Key into every event it sends to BigPanda.
5. BigPanda will generate a page containing setup instructions for the Cribl Stream Webhook Destination. Store the following information:
   - **URL**: `https://integrations.bigpanda.io/oim/cribl/alerts`.
   - **Method**: `POST`.
   - **Format**: `Custom`.
   - **Content Type**: `application/json`.
   - **Authentication token**: A 32-character `<auth-token>`.

The BigPanda page will also contain test code. You'll need the preceding information and the test code when configuring Cribl Stream in the next section.

Configure the Webhook Destination in Cribl Stream

1. From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Destinations**. Then select **Webhook** from the **Manage Destinations** page's tiles or left nav. Click **New Destination** to open the **Webhook** > **New Destination** modal.
2. On the modal's **Configure** > **General Settings** tab, enter or select the following values:
   - **URL**: Enter the Alerts API endpoint URL, for example: `https://integrations.bigpanda.io/oim/cribl/alerts`.
   - **Method**: `POST`.
   - **Format**: `Custom`.

    - **Content type**: `application/json`.
3. In the **Authentication** tab, select an **Authentication type**.
    - You can select **Auth token**, and then enter the Access Token (the 32-character string you wrote down earlier) in the **Token** field.
    - Alternatively, select **Auth token (text secret)** to expose the **Secret** drop-down, in which you can select a stored secret that references the Access Token. A **Create** link is available to store a new, reusable secret.
4. Click **Save**, then **Commit & Deploy**. You are now ready to test your Webhook Destination's communication with Big Panda.
5. In the **Test** tab, enter the test code from the BigPanda setup instructions. It will look like this:

```
[
      {
               "app_key": "<your_app_key>",
               "status": "critical",
               "host": "production-database-1",
               "timestamp": 1402302570,
               "check": "CPU overloaded",
               "description": "CPU is above upper limit (70%)",
               "cluster": "production-databases",
               "my_unique_attribute": "myUniqueValue987654321"
      }
]
```

5. Click **Run Test**.

This should send an alert to BigPanda.

BigPanda Alerts API Requirements

HTTP payloads sent to the BigPanda Alerts API must satisfy rules that are beyond the scope of this topic. For details, see the BigPanda documentation about Alert Properties and Integration Diagnostics.

However, at at minimum, three fields are required:

1. `app_key`.
2. `status`.
3. `host` OR `service` OR `application` OR `device`.

Thus, the test input shown above works even if you omit all but the first three fields.

There are other possibilities for the third field, but they require understanding how BigPanda determines the `primary_property` of an Alert, plus some additional BigPanda configuration. See the BigPanda links above for details.

BigPanda Alert Deduplication

BigPanda processes an alert's Primary and Secondary properties on ingestion. These properties default to `host` and `check`.

When an alert has the same Primary and Secondary properties as those received in previous alerts, the Incident Console shows only the timestamp from the initial event.

For more information on this functionality, see the BigPanda pages Deduplication and Incident_identifier.

Verify that BigPanda is Receiving Notifications and Events

In the BigPanda **Incidents** tab, you should see an Incident whose Source is `Cribl Stream`. The details of the test input you sent from the Webhook Destination should appear in an Alert within that Incident. If so: It works!

17.6.7. System Metrics To Grafana

Cribl Stream 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. While, Grafana is a multi-platform open source analytics and interactive visualization web application. It provides charts, graphs, and alerts for the web when connected to supported data sources.

Cribl has configured a Prometheus dashboard to display Cribl Stream System Metrics, and shared the dashboard in the Grafana library, which is public. This is a relatively simple dashboard suitable for showing aggregate 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.

In this guide, we will show you how to route system metrics to a Grafana dashboard for both basic and highly detailed metrics visualizations.

We will walk you through the following:

- Configure System Metrics Source on Cribl Stream to receive metrics from the host.
- Configure the Prometheus Destination to send data to Grafana Cloud.
- Access Grafana Cloud to see the metrics and visualizations.

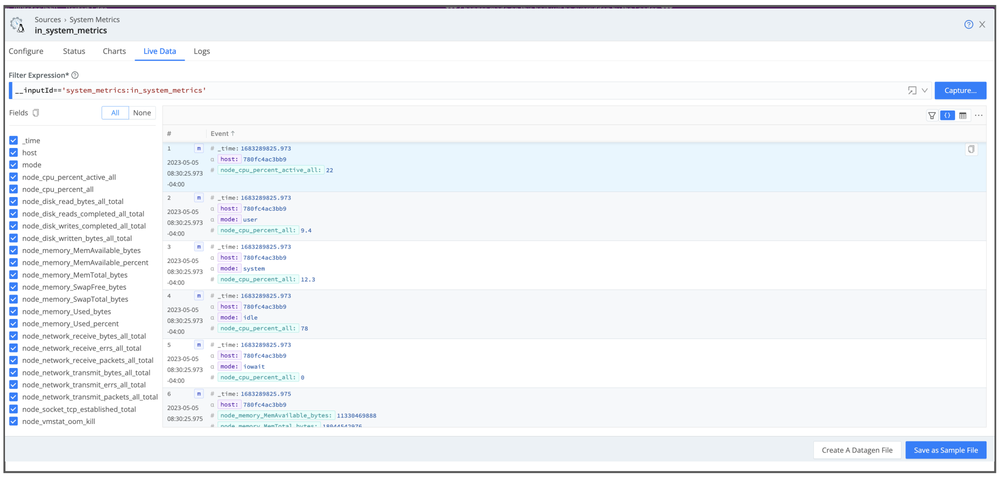
Configure the System Metrics Source on Cribl Stream

In Cribl Stream, start by configuring and enabling a [System Metrics Source](sources-system-metrics. The key requirement here is to to decide on the level of details for the metrics you want to collect. In 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.

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.

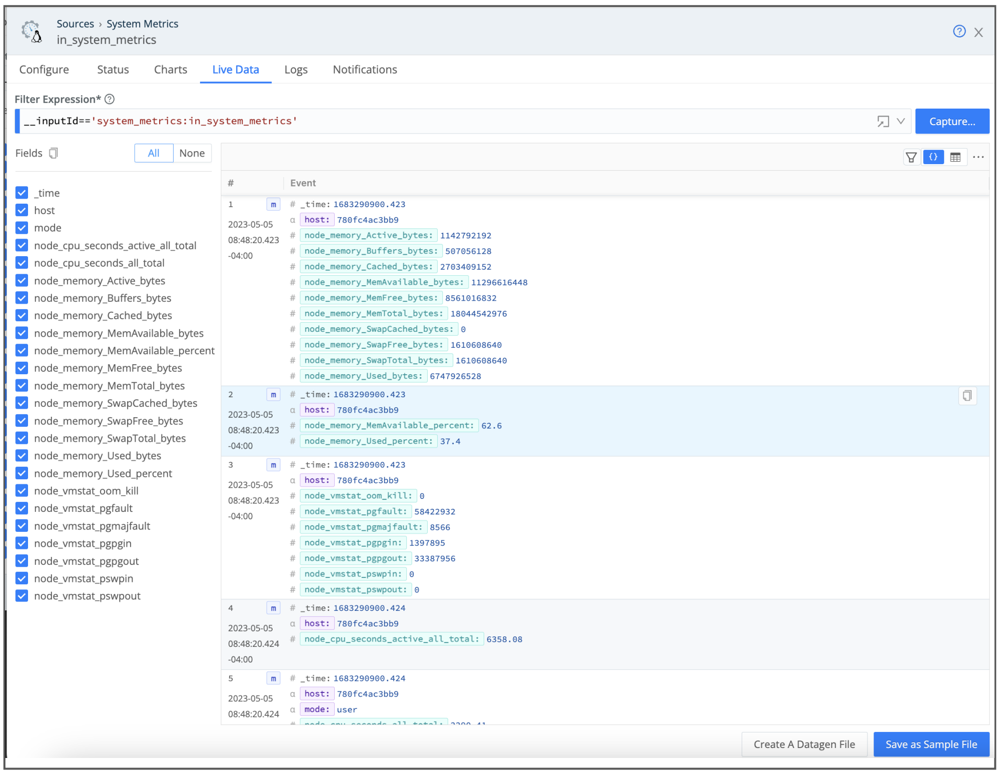


```
__inputId=='system_metrics:in_system_metrics'
```

Fields: _time, host, mode, node_cpu_percent_active_all, node_cpu_percent_all, node_disk_read_bytes_all_total, node_disk_reads_completed_all_total, node_disk_writes_completed_all_total, node_disk_written_bytes_all_total, node_memory_MemAvailable_bytes, node_memory_MemAvailable_percent, node_memory_MemTotal_bytes, node_memory_SwapFree_bytes, node_memory_SwapTotal_bytes, node_memory_Used_bytes, node_memory_Used_percent, node_network_receive_bytes_all_total, node_network_receive_errs_all_total, node_network_receive_packets_all_total, node_network_transmit_bytes_all_total, node_network_transmit_errs_all_total, node_network_transmit_packets_all_total, node_socket_tcp_established_total, node_vmstat_oom_kill

```
#                              Event ↑
1  2023-05-05 08:30:25.973 -04:00
   _time: 1683289825.973
   host: 780fc4ac3bb9
   node_cpu_percent_active_all: 22
2  2023-05-05 08:30:25.973 -04:00
   _time: 1683289825.973
   host: 780fc4ac3bb9
   mode: user
   node_cpu_percent_all: 9.4
3  2023-05-05 08:30:25.973 -04:00
   _time: 1683289825.973
   host: 780fc4ac3bb9
   mode: system
   node_cpu_percent_all: 12.3
4  2023-05-05 08:30:25.973 -04:00
   _time: 1683289825.973
   host: 780fc4ac3bb9
   mode: idle
   node_cpu_percent_all: 78
5  2023-05-05 08:30:25.973 -04:00
   _time: 1683289825.973
   host: 780fc4ac3bb9
   mode: iowait
   node_cpu_percent_all: 0
6  2023-05-05 08:30:25.975
   _time: 1683289825.975
   host: 780fc4ac3bb9
   node_memory_MemAvailable_bytes: 11330469888
   node_memory_MemTotal_bytes: 18044542976
```

Basis System Metrics



```
__inputId=='system_metrics:in_system_metrics'
```

Fields: _time, host, mode, node_cpu_seconds_active_all_total, node_cpu_seconds_all_total, node_memory_Active_bytes, node_memory_Buffers_bytes, node_memory_Cached_bytes, node_memory_MemAvailable_bytes, node_memory_MemAvailable_percent, node_memory_MemFree_bytes, node_memory_MemTotal_bytes, node_memory_SwapCached_bytes, node_memory_SwapFree_bytes, node_memory_SwapTotal_bytes, node_memory_Used_bytes, node_memory_Used_percent, node_vmstat_oom_kill, node_vmstat_pgfault, node_vmstat_pgmajfault, node_vmstat_pgpgin, node_vmstat_pgpgout, node_vmstat_pswpin, node_vmstat_pswpout

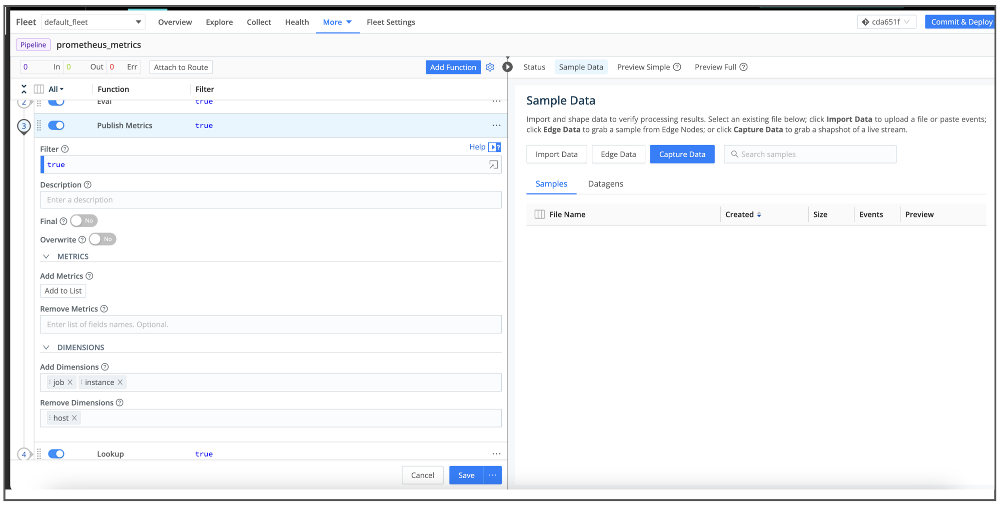
```
#                              Event
1  2023-05-05 08:48:20.423 -04:00
   _time: 1683290900.423
   host: 780fc4ac3bb9
   node_memory_Active_bytes: 1142792192
   node_memory_Buffers_bytes: 507056128
   node_memory_Cached_bytes: 2703409152
   node_memory_MemAvailable_bytes: 11296616448
   node_memory_MemFree_bytes: 8561016832
   node_memory_MemTotal_bytes: 18044542976
   node_memory_SwapCached_bytes: 0
   node_memory_SwapFree_bytes: 1610608640
   node_memory_SwapTotal_bytes: 1610608640
   node_memory_Used_bytes: 6747926528
2  2023-05-05 08:48:20.423 -04:00
   _time: 1683290900.423
   host: 780fc4ac3bb9
   node_memory_MemAvailable_percent: 62.6
   node_memory_Used_percent: 37.4
3  2023-05-05 08:48:20.423 -04:00
   _time: 1683290900.423
   host: 780fc4ac3bb9
   node_vmstat_oom_kill: 0
   node_vmstat_pgfault: 58422932
   node_vmstat_pgmajfault: 8566
   node_vmstat_pgpgin: 1397895
   node_vmstat_pgpgout: 33387956
   node_vmstat_pswpin: 0
   node_vmstat_pswpout: 0
4  2023-05-05 08:48:20.424 -04:00
   _time: 1683290900.424
   host: 780fc4ac3bb9
   node_cpu_seconds_active_all_total: 6358.08
5  2023-05-05 08:48:20.424
   _time: 1683290900.424
   host: 780fc4ac3bb9
   mode: user
```

Detailed System Metrics

(Optional) Adjust the `prometheus_metrics` Pipeline

Optionally, you can adjust the `prometheus_metrics` Pipeline to include the `host` dimension or 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**.


Fleet default_fleet
Overview Explore Collect Health More Fleet Settings
cda651f
Commit & Deploy
Pipeline prometheus_metrics
Add Function
Attach to Route
Status Sample Data Preview Simple Preview Full
All Function Filter
Import Data Edge Data Capture Data
Search samples
Samples Datagens
File Name Created Size Events Preview
Cancel Save

Function	Filter
Eval	true
Publish Metrics	true
Lookup	true

Filter: `true`
Description: Enter a description
Final: No
Overwrite: No
METRICS
Add Metrics: Add to List
Remove Metrics: Enter list of fields names. Optional.
DIMENSIONS
Add Dimensions: job, instance
Remove Dimensions: host

Sample Data

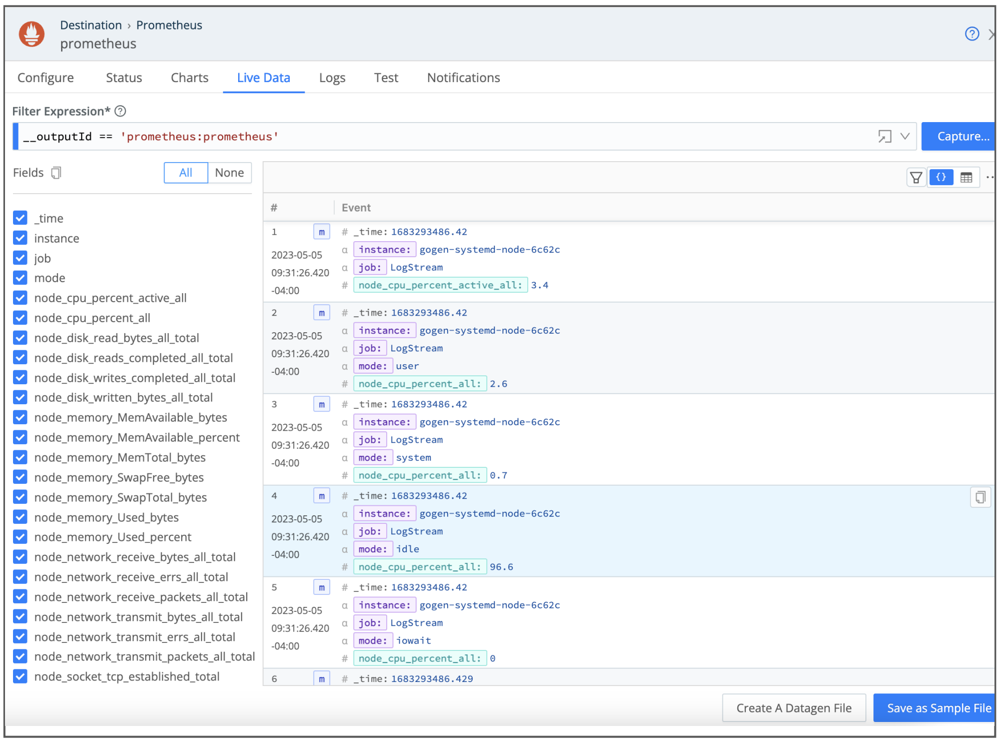
Import and shape data to verify processing results. Select an existing file below; click **Import Data** to upload a file or paste events; click **Edge Data** to grab a sample from Edge Nodes; or click **Capture Data** to grab a snapshot of a live stream.

Adjust the Pipeline

Configure the Prometheus Destination

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

Next, preview your data on the modal's **Live Data** tab.



Filter Expression*

```
__outputId == 'prometheus:prometheus'
```

Fields: _time, instance, job, mode, node_cpu_percent_active_all, node_cpu_percent_all, node_disk_read_bytes_all_total, node_disk_reads_completed_all_total, node_disk_writes_completed_all_total, node_disk_written_bytes_all_total, node_memory_MemAvailable_bytes, node_memory_MemAvailable_percent, node_memory_MemTotal_bytes, node_memory_SwapFree_bytes, node_memory_SwapTotal_bytes, node_memory_Used_bytes, node_memory_Used_percent, node_network_receive_bytes_all_total, node_network_receive_errs_all_total, node_network_receive_packets_all_total, node_network_transmit_bytes_all_total, node_network_transmit_errs_all_total, node_network_transmit_packets_all_total, node_socket_tcp_established_total

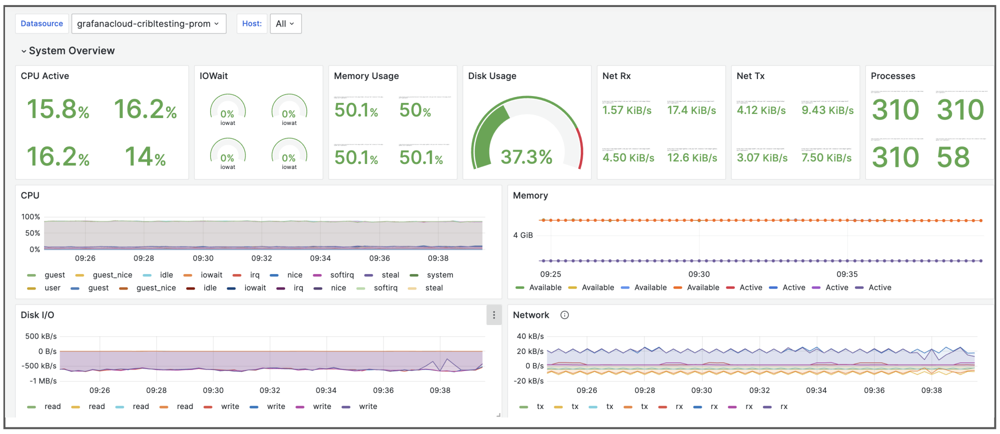
```
#  Event
1  2023-05-05 09:31:26.420 -04:00
   _time: 1683293486.42
   instance: gogen-systemd-node-6c62c
   job: LogStream
   node_cpu_percent_active_all: 3.4
2  2023-05-05 09:31:26.420 -04:00
   _time: 1683293486.42
   instance: gogen-systemd-node-6c62c
   job: LogStream
   mode: user
   node_cpu_percent_all: 2.6
3  2023-05-05 09:31:26.420 -04:00
   _time: 1683293486.42
   instance: gogen-systemd-node-6c62c
   job: LogStream
   mode: system
   node_cpu_percent_all: 0.7
4  2023-05-05 09:31:26.420 -04:00
   _time: 1683293486.42
   instance: gogen-systemd-node-6c62c
   job: LogStream
   mode: idle
   node_cpu_percent_all: 96.6
5  2023-05-05 09:31:26.420 -04:00
   _time: 1683293486.42
   instance: gogen-systemd-node-6c62c
   job: LogStream
   mode: iowait
   node_cpu_percent_all: 0
6  _time: 1683293486.429
```

Preview Live Data

Visualize the Data in Grafana

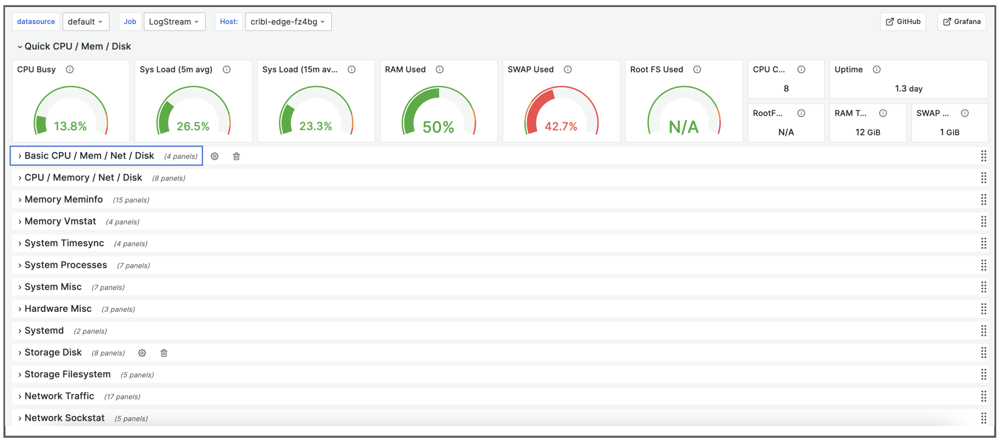
Navigate to your Grafana Cloud instance, and select **Explore** on the left panel to review the node metrics. Selecting metrics and running the query will produce a chart and table of metric events that can be added to new or existing dashboards for analysis. If you are missing some expected metrics, you might need to adjust the variables in your Grafana Cloud instance. For details, see Variables.

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



Cribl Node Observability

To view the detailed panels, select the arrow for each category:



Node Exporter Full

17.6.8. Kafka Authentication With Kerberos

This page describes how to set up a Kafka Source and Destination configured to use Kerberos authentication. It also documents the verification steps required to test your setup.

Kerberos is only available on Linux, so you'll have to use a Linux host or Docker container to work through this procedure.

For additional information on Kafka configuration, see also Kafka Sources or Kafka Destinations.

Setting up the Cribl Stream Host

Now that you have set up an instance of Kafka with Kerberos, perform the following steps on the same host that Stream is running on.

1. Install `krb5-user` (or the equivalent for non-Ubuntu systems) on your Worker Node.
2. Ensure you have a valid `/etc/krb5.conf` file. Consult your Kerberos admin if you don't have one. Kerberos authentication with Kafka specifically requires `rdns = false` in this config, but no other special settings. Cribl Stream supports the file and directory cache types.
3. Ensure the keytab path in your configuration points to a valid keytab on each Worker Node.

Configuring the Kafka Destination

Create a new Kafka Destination with the following settings:

General Settings

- **Brokers**: `broker.kerberos-demo.local:9092`
- **Topic**: `Cribl`

Authentication Settings

- **Enabled**: `Yes`
- **SASL mechanism**: `GSSAPI/Kerberos`
- **Keytab location**: `/<path-to-client-keytab>/kafka-client.key`. This path to the valid keytab file must be available on all Worker Nodes.
- **Principal**: `kafka_producer@TEST.CONFLUENT.IO`
- **Broker service class**: `kafka`

Configuring the Kafka Source

Create a new Kafka Source with the following settings:

General Settings

- **Brokers**: `broker.kerberos-demo.local:9092`
- **Topic**: `Cribl`
- **Group ID**: `Cribl`

Authentication Settings

- **Enabled**: `Yes`
- **SASL mechanism**: `GSSAPI/Kerberos`
- **Keytab location**: `/<path-to-client-keytab>/kafka-client.key`
- **Principal**: `kafka_producer@TEST.CONFLUENT.IO`
- **Broker service class**: `kafka`

Verifying the Setup

1. Open the **Live Data** tab and start a capture with **Capture Time (sec)** and **Capture Up to N Events** set to high values (e.g., `1000`).
2. In a separate browser tab, navigate to the Destination's **Test** tab and click **Run Test** a few times.
3. Return to the Source's **Live Data** tab and verify that the Source is capturing events.
4. When you're done testing, run `./stop.sh` from the same directory where you ran `./start.sh` to set up your environment.

17.6.9. Moogsoft/Webhook Integration

Written for Cribl by Douglas Bothwell, formerly of Moogsoft.

You can configure Cribl Stream to send Webhook notifications to Moogsoft, using custom integrations. A custom integration is a user-defined Moogsoft endpoint that ingests JSON payloads, and converts them to Moogsoft events or metrics.

Note these options:

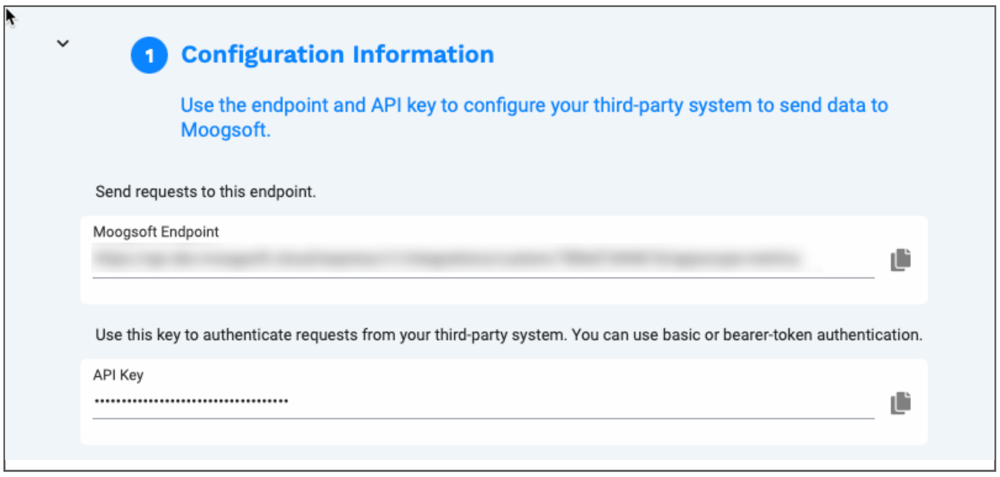
- A Moogsoft custom integration can ingest either events or metrics. Each custom integration has its own separate API key, and its own Cribl-to-Moogsoft mappings.
- There is no limit to the number of custom integrations you can create.
- The examples below illustrate simple event and metric mappings. You can define your own mappings for each endpoint, based on the data types and payloads you want to send. For details, see Moogsoft's Create Your Own Integration documentation.

Create the Moogsoft Custom Integration

Log into your Moogsoft SaaS instance as an Owner or Administrator, and then:

1. Choose **Data Config** > **Ingestion Services** > **Create Your Own Integration**.
2. Click **Add New Integration**.
3. Specify an integration endpoint and description.
4. Specify the data type to send to the endpoint: **Events** or **Metrics**.

The integration setup screen appears, with the URL and the API key for the new endpoint.



1 Configuration Information

Use the endpoint and API key to configure your third-party system to send data to Moogsoft.

Send requests to this endpoint.

Moogsoft Endpoint
[redacted]

Use this key to authenticate requests from your third-party system. You can use basic or bearer-token authentication.

API Key
••••••••••••••••••••••••••••••••••••

The Moogsoft integration setup screen

Configure the Webhook Destination

From a Cribl Stream instance's or Group's **Manage** submenu, select **Data** > **Destinations**, then select **Webhook** from the **Manage Destinations** page's tiles or left nav. Click **Add New** to open the **Webhook** > **New Destination** modal, and complete the following options and fields.

General Settings

URL: The URL for the new custom integration (copy this from the Moogsoft UI).

Method: `POST`.

Format: `Custom`.

Content type: `application/json`.

Advanced Settings

Extra HTTP Headers: Click **Add Header** to define this extra header:

Name	Value
`apiKey`	The API key for the new custom integration (copy this from the Moogsoft UI).

Click **Save**, then **Commit** and **Deploy**. Now you are ready to map your events and/or metrics.

Mapping Cribl Data to Moogsoft

Your mappings will differ, depending on the data you want to send. However, the simple examples below, in which we map a Cribl Stream payload to a Moogsoft custom integration, illustrate principles that apply to all custom mappings:

- Moogsoft has a defined event schema and metric schema. Each schema includes a set of required fields. Your custom integration must include mappings for all required fields.
- You can define custom tags for Cribl Stream fields that do not have Moogsoft equivalents.
- You can also specify default values in case Cribl Stream sends an object with a missing field.

- The Moogsoft event schema includes a `severity` field. You can map Cribl Stream fields and values to Moogsoft severities, or define a default severity if a payload does not include this information.

Define and validate your mappings based on the type of data you plan to send to Moogsoft:

- Event Mapping
- Metric Mapping

Event Mapping

To illustrate how to map Cribl Stream payloads to Moogsoft events, we'll walk through an example with a payload of Cribl Stream syslog messages. You'll follow the same overall workflow for any events payload.

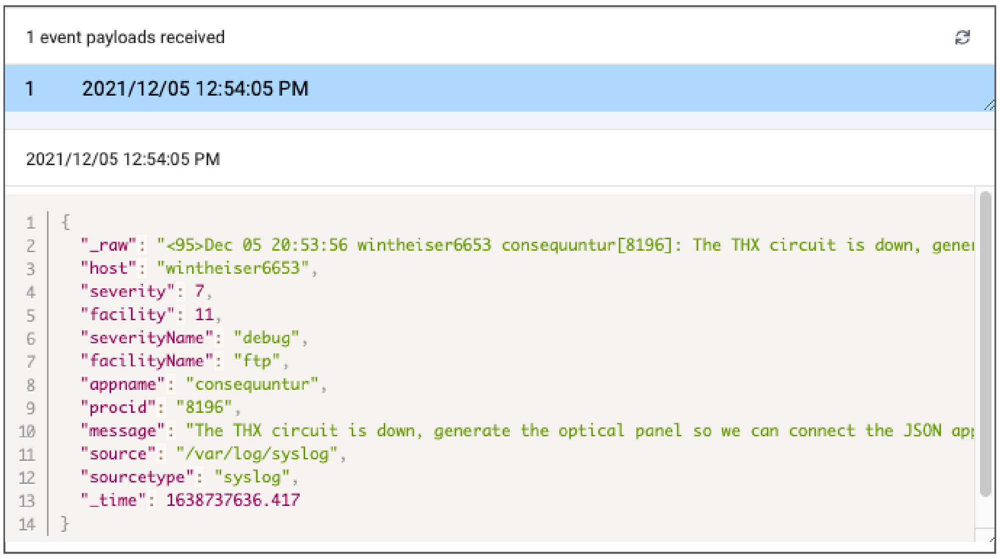
Send a Sample Payload to Moogsoft

In Cribl Stream, navigate to your Webhook Destination's configuration modal, click the **Test** tab, then:

1. In the **Test input** field, define one or more JSON payloads for the Cribl data you want to send. To map syslog events, set the **Select sample** drop-down to **syslog.log**.
2. Click **Test**.

Map the Event Fields

In Moogsoft, view the custom integration's config screen. Under **Map Your Data**, you should now see the payload you just sent.



```
{
  "_raw": "<95>Dec 05 20:53:56 wintheiser6653 consequuntur[8196]: The THX circuit is down, gener
  "host": "wintheiser6653",
  "severity": 7,
  "facility": 11,
  "severityName": "debug",
  "facilityName": "ftp",
  "appname": "consequuntur",
  "procid": "8196",
  "message": "The THX circuit is down, generate the optical panel so we can connect the JSON ap
  "source": "/var/log/syslog",
  "sourcetype": "syslog",
  "_time": 1638737636.417
}
```

The event payload appears in Moogsoft

Select the payload, and then define your Cribl-to-Moogsoft mappings. Your Cribl Stream data will largely determine the mappings you want. See Events Object in the Moogsoft API docs.

Here are some reasonable mappings for the syslog payload in this example:

Cribl Source Field(s)	Moogsoft Target Field
`message`	`description`
`appname`	`service`
`facilityName`	`check`
`severity`, `severityName`	`severity`
`procid`	`tag.process-id`

Map the Severities

The Moogsoft event schema has a severity field. You can specify integers or strings, from `0`, meaning Clear, to `5`, meaning Critical. Here are some reasonable mappings.

Syslog Severities	Moogsoft Severities
Emergency (`0`), Alert (`1`), Critical (`2`)	Critical (`5`)
Error (`3`)	Major (`4`)
Warning (`4`)	Warning (`2`)
Informational (`6`), Debug (`7`)	Unknown (`1`)
Notice (`5`)	Clear (`0`)

Verify your Mappings

Once you save and apply your mappings in Moogsoft, do the following:

1. In Cribl Stream, return to the **Test** tab for the Webhook Destination. Click **Test** again to send another payload.
2. In Moogsoft, go to the **Alerts** screen. You should now see a new alert based on the payload you just sent.

Metric Mapping

To illustrate how to map Cribl Stream payloads to Moogsoft events, we'll walk through an example with a payload of Cribl Stream syslog messages. You'll follow the same overall workflow for any metrics payload.

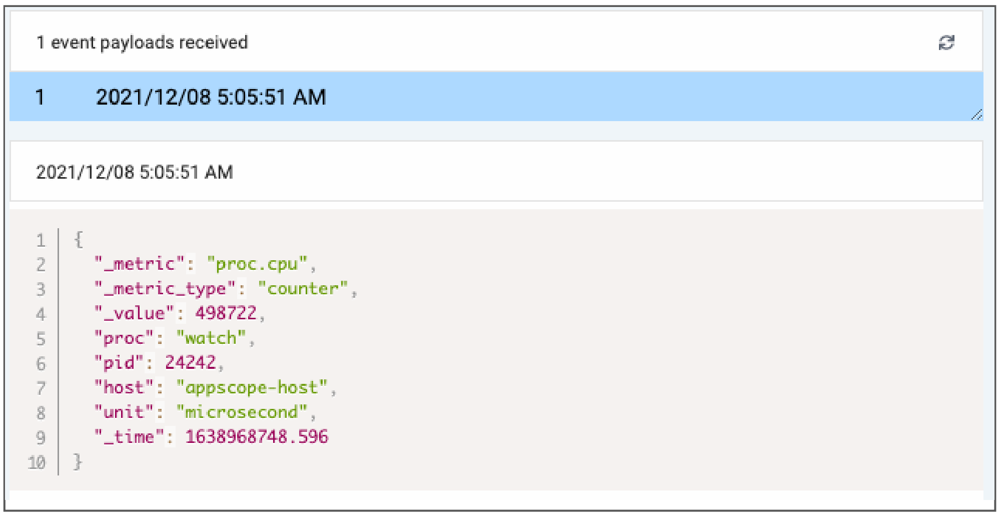
Send a Sample Payload to Moogsoft

In Cribl Stream, navigate to your Webhook Destination's configuration modal, click the **Test** tab, then:

1. In the **Test input** field, define one or more JSON payloads for the Cribl data you want to send. To map syslog events, set the **Select sample** drop-down to **appscope-metrics.log**.
2. Click **Test**.

Map the Metrics Fields

In Moogsoft, view the custom integration's config screen. Under **Map Your Data**, you should now see the payload you just sent.



```
{
  "_metric": "proc.cpu",
  "_metric_type": "counter",
  "_value": 498722,
  "proc": "watch",
  "pid": 24242,
  "host": "appscope-host",
  "unit": "microsecond",
  "_time": 1638968748.596
}
```

The metrics payload appears in Moogsoft

Select the payload and then define the Cribl-to-Moogsoft mappings you want. Your Cribl data will largely determine these mappings. See Metric Datum Object in the Moogsoft API docs.

Here are some reasonable mappings for the AppScope payload you just sent:

Cribl Source Fields	Moogsoft Target Field
`_metric`	`metric`
`_value`	`data`
`host`	`source`
`unit`	`tag.unit`

Cribl Source Fields	Moogsoft Target Field
`pid`	`tag.pid`

Verify Your Mappings

After you save and apply your mappings in Moogsoft, test them like this:

1. In Cribl Stream, return to the Webhook Destination’s **Test** tab. Click **Test** again to send another payload.
2. In Moogsoft, go to the **Metrics** screen. You should now see a new alert, based on the payload you just sent.

17.6.10. Nightfall Integration

Thanks to Nightfall for contributing this page and the corresponding Nightfall Pack, which you can install in Cribl Stream and customize as described below.

The Nightfall Pack relies on Nightfall's Data Loss Prevention (DLP) engine, which uses machine learning to detect sensitive information in events streamed through Cribl. The Pack enables you to monitor your Pipelines in real time for PII, PHI, PCI, and other sensitive data. Nightfall discovers and automatically redacts the sensitive content before it can reach any Destination. Aside from redaction, you can use the Pack to automatically label problematic event data.

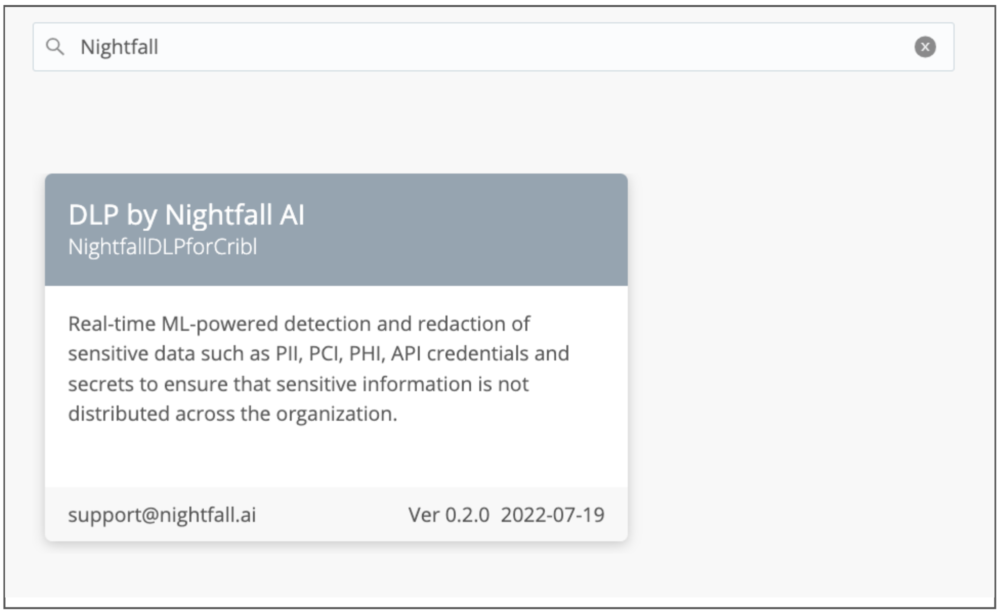
The risk of sensitive data spreading across data pipelines grows as organizations work with data from more sources. Nightfall can help ensure that sensitive data doesn't accumulate in the first place, facilitating compliance with data-security regulations. This is important for Cloud-first organizations that must scale observability while complying with regional or industry-specific compliance regimes.

Configuring the Detection Engine

1. Sign up for a free Nightfall Developer Platform account, if you have not already done so. This entitles you to scan up to 3GB of event data per month for free.
2. Log in to the Nightfall Dashboard.
3. Create an API Key.
4. Create the Detection Rules and/or Policy that you want Nightfall to scan with.
   Detection Rules enable you to specify which Detectors will scan each event. A Detection Rule simply redacts sensitive information from events.
   A **Policy** governs alerts, which Nightfall can send via email, Slack, or webhook. (Assuming that you want to send data to a Cribl Stream Destination and/or a SIEM, use a webhook.) A Policy can include up to 20 previously defined Detection Rules. It can also turn redaction off, if desired.

Installing the Nightfall Pack

1. From Cribl Stream's top nav, select **Processing** > **Packs**.
2. Click **Add Pack**, then select **Add from Dispensary** to open the Packs Dispensary drawer.
3. Navigate to the **DLP by Nighfall AI** Pack's tile.


Nightfall

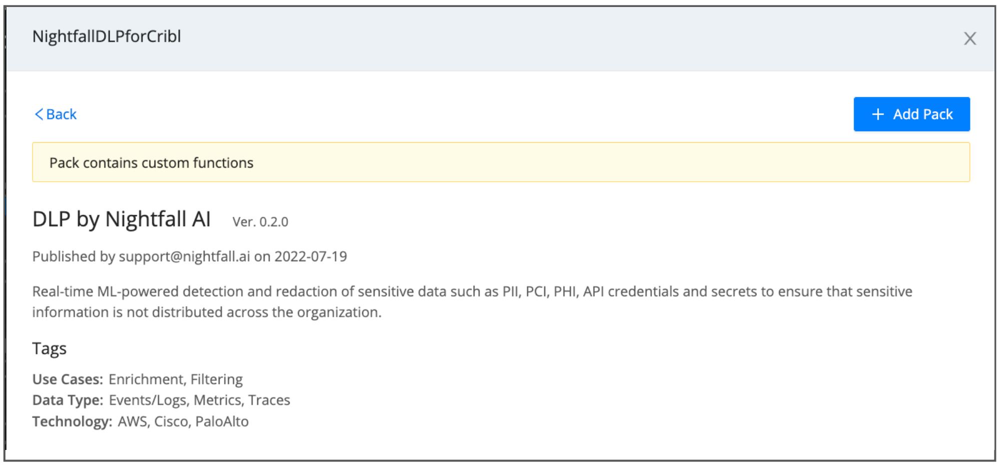
DLP by Nightfall AI
NightfallDLPforCribl

Real-time ML-powered detection and redaction of sensitive data such as PII, PCI, PHI, API credentials and secrets to ensure that sensitive information is not distributed across the organization.

support@nightfall.ai Ver 0.2.0 2022-07-19

Locating the Nightfall Pack

1. Click the Nightfall Pack's tile to display its details page.


NightfallDLPforCribl

Pack contains custom functions

DLP by Nightfall AI Ver. 0.2.0

Published by support@nightfall.ai on 2022-07-19

Real-time ML-powered detection and redaction of sensitive data such as PII, PCI, PHI, API credentials and secrets to ensure that sensitive information is not distributed across the organization.

Tags

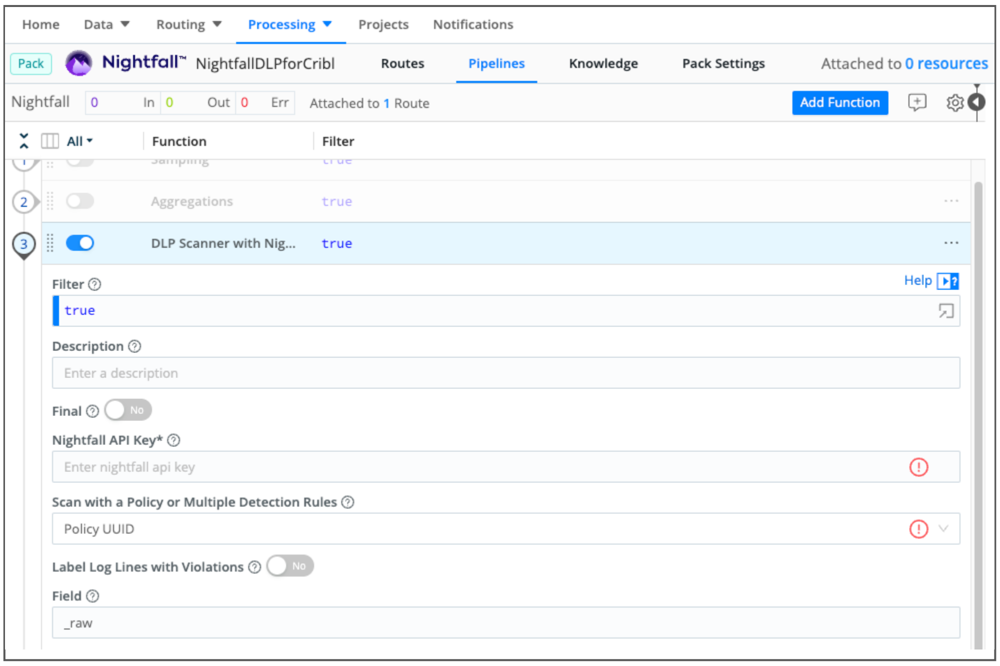
Use Cases: Enrichment, Filtering
Data Type: Events/Logs, Metrics, Traces
Technology: AWS, Cisco, PaloAlto

The Nightfall Pack details page

1. Click **Add Pack**. Shortly, you'll see a banner confirming that the Pack is installed.

Configuring the Nightfall Pack

1. From the **Manage Packs** page, click the new Pack. This opens the Pack's **Routes** tab.
2. In the **Pipeline** column, click `Nightfall`.
3. From the resulting Nightfall Pipeline configuration, expand the **DLP Scanner with Nightfall AI** Function.



Nightfall 0 In 0 Out 0 Err Attached to 1 Route

	Function	Filter
1	Sampling	true
2	Aggregations	true
3	DLP Scanner with Nig...	true

Filter: true

Description: Enter a description

Final: No

Nightfall API Key*: Enter nightfall api key

Scan with a Policy or Multiple Detection Rules: Policy UUID

Label Log Lines with Violations: No

Field: _raw

Selecting the Nightfall Pipeline

1. In the **Nightfall API Key** field, enter the API key you created in the Nightfall Dashboard.
2. Set the **Scan with a Policy or Multiple Detection Rules** drop-down to match your detection engine configuration.
   Selecting **Policy UUID** exposes a corresponding field to enter the single UUID.
   Selecting **Detection Rule UUID** exposes a **Add Rule**. Click this to enter as many Detection Rule UUIDs as you want to specify.
3. (Optionally:) Label sensitive events. If you toggle **Label Log Lines with nightfall_violations** to `Yes`, each event where Nightfall has detected sensitive information will have an added `nightfall_violations: true` field.

Advanced Features

The Nightfall Pipeline also includes three limiting Functions, described here. These are disabled by default, but you can enable them as needed.

Sampling

To reduce the data usage that's billed against your Nightfall plan, you can enable the Sampling Function. This way, Nightfall will scan only a subset of each streamed file, instead of every event.

Log Batching

To reduce the number of requests that Cribl Stream makes to the Nightfall API, you can enable log batching. To do so, enable both the Aggregations and Unroll Functions, then configure the Aggregation Function's

Time window.

The Nightfall DLP plugin will automatically detect the aggregations, and unroll the batches, only after Cribl Stream sends them to the Detection Engine. Then, Nightfall will process each event separately, as it normally does.

17.6.11. Managing QRadar Licenses

If you're using a SIEM (security information and event management) system like QRadar that's subject to an events-per-second (EPS) licensing model, managing license costs can be an issue. Cribl Stream enables you to proactively manage your license by slowing EPS growth.

Cribl Stream does so by offering fine-grained control over what events to send to the QRadar Event Processor, without breaking the expected LEEF (Log Event Extended Format) data format. You can report on event codes, and make decisions about what to drop, based on the data in the events. Some of what we discuss here also applies to SIEMs generally, not just QRadar.

Confronting QRadar Constraints

QRadar imposes constraints on data through several of its components:

- QRadar Event Collectors can send data only to QRadar. Elastic and other systems of retention or analysis are not an option.
- QRadar Event Collectors can send data only in LEEF format. The pipeline cannot provide additional streams of data in different formats such as JSON or syslog.
- Incoming data must adhere to a predefined schema, based on IBM's LEEF format. You can't reformat events, nor reduce the size of an event, without breaking the correlation rules within the QRadar Event Processor.
- When incoming data exceeds the EPS license limits, QRadar will drop or stop ingesting data. Meanwhile, since the license is based on EPS event velocity, license costs increase linearly: the higher the EPS count, the higher the cost.

If your event velocity approaches the license limit, your options are unappealing:

- You can try to exclude some data, but the correlation rules and constraints on data formatting restrict your ability to be selective. You can drop only broad categories of data, such as an entire Windows Event ID, leaving you at a risk of missing something important.
- You can hope that you never actually exceed the EPS limit, but if you guess wrong, QRadar dropping or not ingesting data can be disruptive and compromise your enterprise security posture.

Cribl Stream enables you to surgically minimize these risks, and get the most out of your existing licenses.

Dropping Events Selectively

A common tactic to avoid exceeding an EPS license limit is to start dropping an entire class of events. This is a crude approach that carries its own risks. With Cribl Stream, you can analyze the data within the event, and decide what to keep based on your priorities.

Analyzing a Windows Event

Here's an example of how to analyze and selectively drop Windows Security Event ID 4674. This event documents operations attempted on a privileged object. While it's commonly discarded when admins are feeling pressed to drop events, 4674 is actually a critical ID that should be kept some, or most, of the time.

Event ID 4674 includes the logon types shown in the table below.

Logon Type	Logon Title	Description
2	`Interactive`	Activity on the system's keyboard and screen.
3	`Network`	Connection to a shared folder on this computer from elsewhere on the network.
4	`Batch`	Processes might be executing on behalf of a user without their direct intervention.
5	`Service`	Service startup by the Service Control Manager.
7	`Unlock`	An unattended workstation with password-protected screen saver.
8	`NetworkCleartext`	Logon with credentials sent in cleartext. Most often indicates a logon to IIS with "basic authentication".
9	`RemoteInteractive`	Logon with RunAs, or mapping a network drive with alternate credentials.
10	`CachedInteractive`	Logon with cached domain credentials, such as when logging on to a laptop when away from the network.

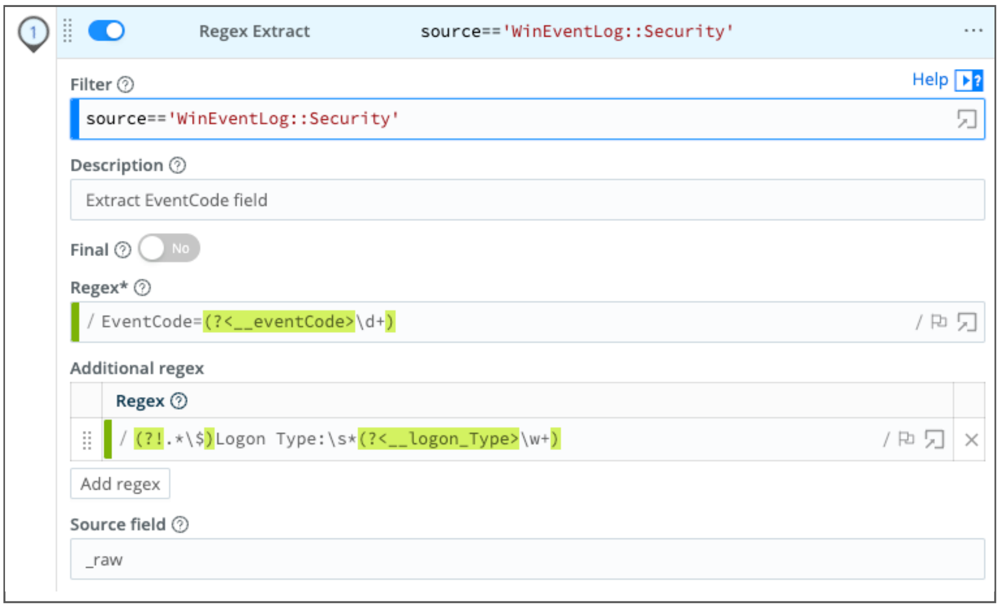
Each logon type is assigned a different security value.

Let's suppose that your QRadar deployment is in an enterprise where inbound file sharing is disabled. In that case, you don't need logon type 3 events. So let's see how to use Cribl Stream to keep all ID 4674 events except logon type 3 events, which we'll' drop.

Dropping the Excess Baggage

Use sample data to validate the reduction methods described below. Only when you are sure they won't interfere with your system's usability should you apply these methods to your production SIEM.

1. Save your Windows Events to Cribl Stream using Live Capture.
2. Create a Pipeline and name it `Windows_Event_Reduction`.
3. Add the Regex Extract Function to your Pipeline to create fields for `EventCode` and `Logon Type`. These fields will be the foundation for a Filter that drops only Windows events of ID 4674, logon type 3.



Regex Extract source=='WinEventLog::Security'

Filter: `source=='WinEventLog::Security'`

Description: Extract EventCode field

Final: No

Regex*: `/ EventCode=(?<__eventCode>\d+)`

Additional regex

Regex: `/ (?!.*\$)Logon Type:\s*(?<__logon_Type>\w+)`

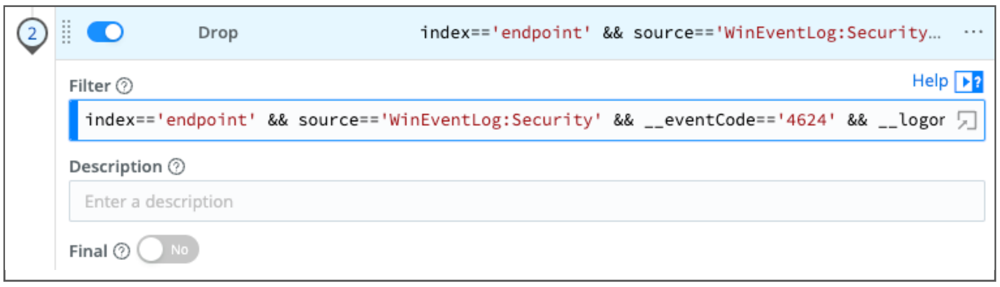
Add regex

Source field: _raw

Regex Function on Pipeline

4. Add the Drop Function to your Pipeline using the Filter below.

```
index=='endpoint' && source=='WinEventLog:Security' && __eventCode=='4624' && __lo
```



Drop index=='endpoint' && source=='WinEventLog:Security...

Filter: `index=='endpoint' && source=='WinEventLog:Security' && __eventCode=='4624' && __logor`

Description: Enter a description

Final: No

Drop Function

5. Apply the Pipeline to your Windows Sources. Cribl Stream will then start dropping the unneeded events.

Other Reduction Examples

Apply the options in this section only after careful analysis of your own environment and priorities. Even if you choose to exclude these events from QRadar, you might want to send them to an object store for later auditing and/or replay.

WinEvents

- Event ID 4624: Look for Subtypes (Logon Type) that can be dropped.
- Event ID's 4634 and 4674: An account was logged off. Most security use cases don't require Logoff events.
- Event ID 4673: A privileged service was called. This event rarely contains actionable intel. It's a large-volume data source that can be dropped, in exchange for other, more-valuable event IDs.
- Event ID 4674: An operation was attempted on a privileged object. Similar to Event ID 4673, this event is largely considered "noise," providing little value. Consider dropping.
- Event ID 4689: A process was exited. This source, when paired with Event ID 4688, can tell you "how long a process was running." Typically, this is unimportant information, and can be omitted from your event collection.

Palo Alto

The following filters can be helpful to reduce the noise:

- All external-to-internal traffic AND `action=deny`: `Ingress traffic && action=deny`
- `PAN Traffic logs && application=ping|ibm-bigfix|outlook-web-online|google-base|traceroute|ms-teams|ic mp|outlook-web-online|ms-office365-base|crowdstrike`

Cisco ASA

- All external-to-internal traffic AND `action=deny`: `Ingress traffic && action=deny`
- Teardown UDP Connection && Event ID 302016
- Teardown TCP connection && Event ID 302014
- Teardown ICMP connection && Event ID 302021
- Teardown Translation && Event ID 305010|305012

17.6.12. Slack/Webhook Integration

You can configure Cribl Stream to send simple Notifications to Slack via a Webhook URL. These Notifications arrive in Slack as messages in a channel that you choose. For more information, see our Slack Notifications topic.

However, if you have specific event data such as alert-based or conformance data that you want to route to Slack for prompt attention, you can use Cribl Stream's Webhook Destination to route or filter this data.

You can also enhance your messages with more advanced formatting. Consult the Slack documentation for detailed information.

Cribl Stream does not currently support the use of special characters in Slack notifications sent via a Webhook Destination.

Creating a Slack/Webhook integration is a three-step process:

1. Create a Slack demo app.
2. Configure the Webhook Destination.
3. Verify that Slack is receiving messages.

Create a Slack Demo App

1. Create a Slack app by following these Slack instructions on sending messages with incoming webhooks.
2. Test the cURL command supplied by the app setup. For example:

```
curl -X POST -H 'Content-type: application/json' --data '{"text":"Hello, World!"}'
```

Configure a Webhook Destination

Configure a Webhook Destination in Cribl Stream as described in Webhook. However, you must override some of the default settings:

- In **General Settings**, use the **URL** provided for your Slack demo app.
- In **General Settings** > **Optional Settings**, set **Format** to `Custom`.

- In **General Settings** > **Optional Settings**, set **Source expression** to `` `{"text":"${_raw}"}` `` (with literal backticks).
- In **Advanced Settings**, set:
  - **Validate server certs** to `No`.
  - **Compress** to `No`.
  - **Max events per request** to `1`.

Verify That Slack Is Receiving Messages

To verify that Slack is receiving message, send a test message, using the **Test** UI. For example:

```
[
  {
    "_raw": "Hello, World!"
  }
]
```

If the integration is working correctly, this test message will arrive in the Slack channel you're using for your demo app.

17.6.13. Tanium To Cribl Stream

Tanium is a security platform that provides rapid searches across multiple endpoints. Tanium integrates with a finite number of tools using its Tanium Connect module, but this can still leave gaps in conforming Tanium output to the specific data formats required by unsupported tools and destinations.

Cribl Stream can help collect, reduce, enrich, transform, and route data from Tanium to any destination. This includes SIEM (Security Information and Event Management) tools, logging tools, or other analytics platforms. In this guide, we'll explain how to configure Tanium Connect to send Tanium-captured data to Cribl Stream. For further details, see Tanium's Configuring SIEM Destinations topic.

Configure Tanium Connect

A Tanium Connection is essentially a scheduled search/collection of data linked to a destination. Tanium translates queries into **Questions**. It formats the Questions into the Tanium Search Language for a search, providing near real-time results.

The first step in configuring Tanium Connect is to set up a Question. A Question can be a Saved Search, Question Log, Client Status, or an Event.

To configure a new connection, go to the **Tanium Module** page and click **Tanium Connect**. On the **Connect Overview** page, scroll to the **Connections** section, and click **Create Connection**.

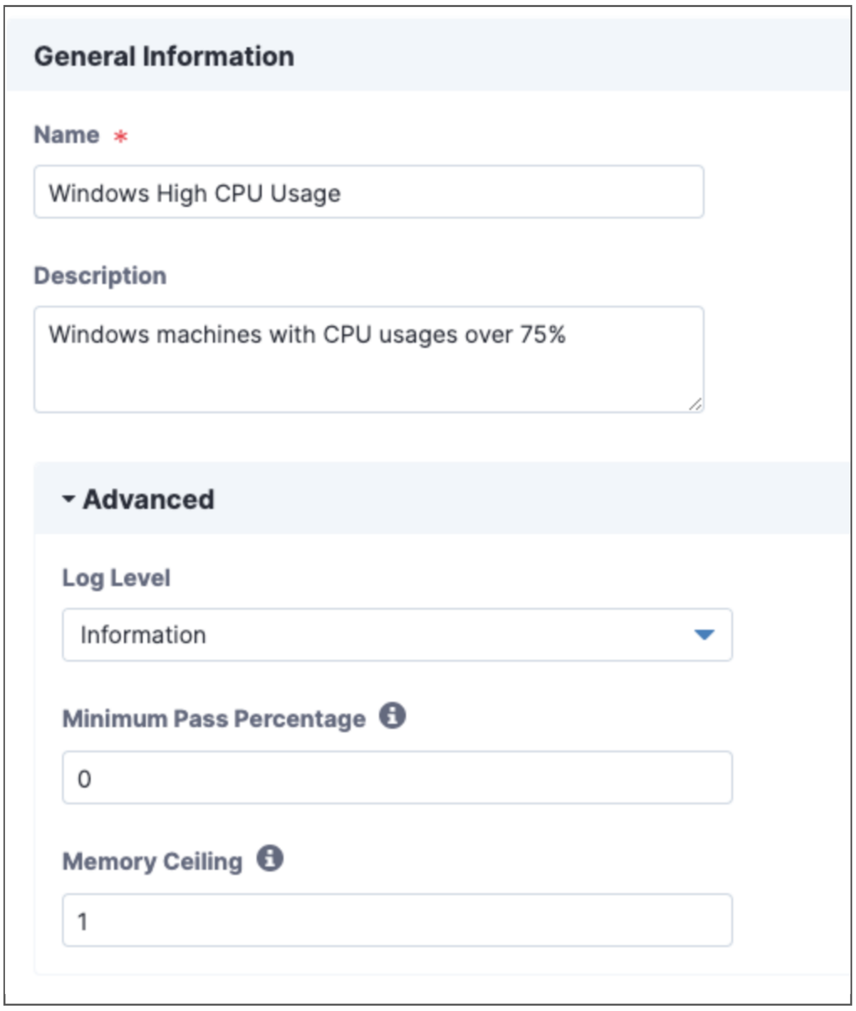
Specify General Connection Information

Name: A unique name for your connection.

Description: An optional description for this connection.

Advanced Settings: Optionally, configure the following fields:

- **Log Level**: Defaults to **Information**. Change the **Log Level** to **Trace** or **Debug** if plan to debug the connection. Alternatively, set the **Log Level** to **Warning**, **Error**, or **Fatal** to reduce the amount of logging.
- **Minimum Pass Percentage**: Minimum percentage of the expected rows to process for the connection to succeed.
- **Memory Ceiling (GB)**: Maximum memory for the node process to run the connection.


General Information

Name *

Windows High CPU Usage

Description

Windows machines with CPU usages over 75%

Advanced

Log Level

Information

Minimum Pass Percentage

0

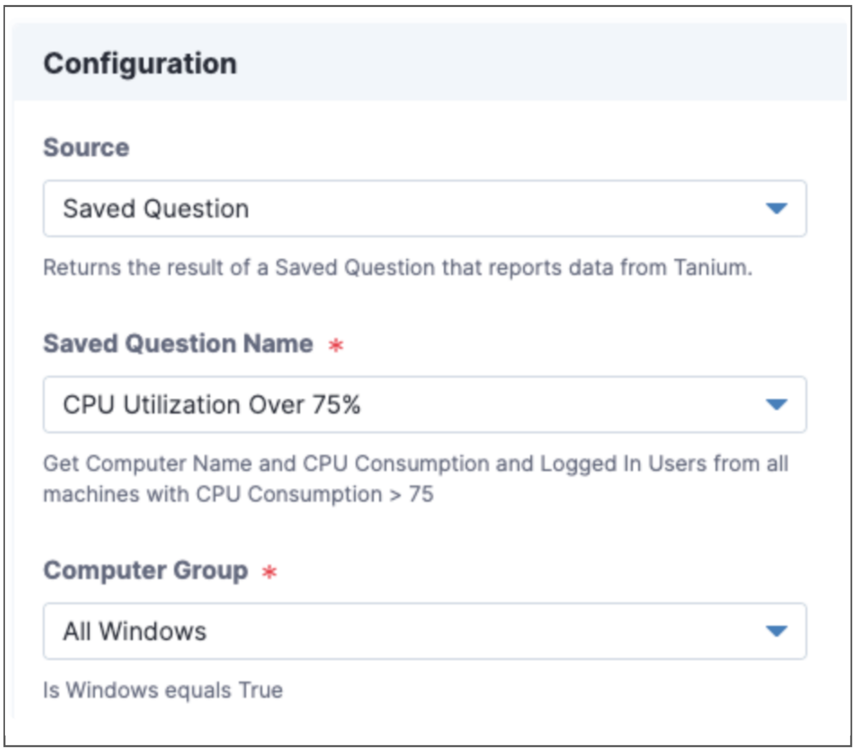
Memory Ceiling

1

General Connection Information tab

Configure the Connection Source

This section enables you to specify the type of data you are sending to your destination. The data is usually information from Tanium, such as a Saved Question, Question Log, Client Status, or Event. The settings vary depending on the Source.


Configuration

Source

Saved Question

Returns the result of a Saved Question that reports data from Tanium.

Saved Question Name *

CPU Utilization Over 75%

Get Computer Name and CPU Consumption and Logged In Users from all machines with CPU Consumption > 75

Computer Group *

All Windows

Is Windows equals True

Connection Source Configuration

When sending logs to Cribl Stream, you must create a network egress rule in Tanium for port 10060. See Tanium's Configure a new network egress rule topic.

Configure Your Destination

In this section, we'll configure Cribl Stream as the destination, using the following fields.

Destination: Select the destination type. For our example, we'll configure it as a `Socket Receiver`.

New: Configure a new destination.

Name: Specify a unique name for your new destination. (This field is displayed only when configuring a **new** destination.)

Existing: Update the settings on an existing destination. (Note that this will affect all the connections that use this destination.)

Destination Name: Drop-down list where you can specify a preconfigured destination. Displayed only when updating an **existing** destination.

Copy Settings: Copy settings from a preconfigured destination.

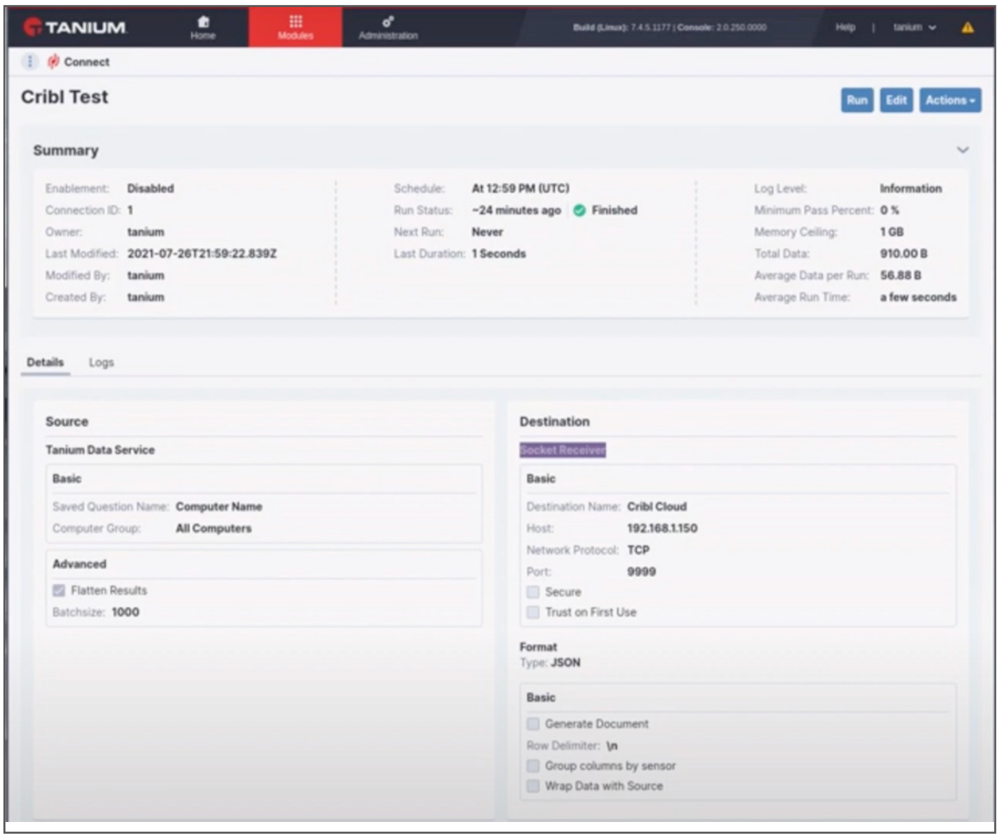
Host: Specify the destination'server host.

Network Protocol: Specify how to connect to the server (e.g., `TCP`).

Port: Enter the port number to listen on.

Secure: Select this option to use TLS encryption.

Trust on First Use: Select this option to accept the certificate presented from the server, and to trust **only** that certificate for future connection runs.



Cribl Test

Summary

Enablement: Disabled
Connection ID: 1
Owner: tanium
Last Modified: 2021-07-26T21:59:22.839Z
Modified By: tanium
Created By: tanium

Schedule: At 12:59 PM (UTC)
Run Status: ~24 minutes ago ✓ Finished
Next Run: Never
Last Duration: 1 Seconds

Log Level: Information
Minimum Pass Percent: 0 %
Memory Ceiling: 1 GB
Total Data: 910.00 B
Average Data per Run: 56.88 B
Average Run Time: a few seconds

Source

Tanium Data Service

Basic

Saved Question Name: Computer Name
Computer Group: All Computers

Advanced

☑ Flatten Results
Batchsize: 1000

Destination

Socket Receiver

Basic

Destination Name: Cribl Cloud
Host: 192.168.1.150
Network Protocol: TCP
Port: 9999
☐ Secure
☐ Trust on First Use

Format
Type: JSON

Basic

☐ Generate Document
Row Delimiter: \n
☐ Group columns by sensor
☐ Wrap Data with Source

Sample Cribl Stream Configured Destination

Format the Data

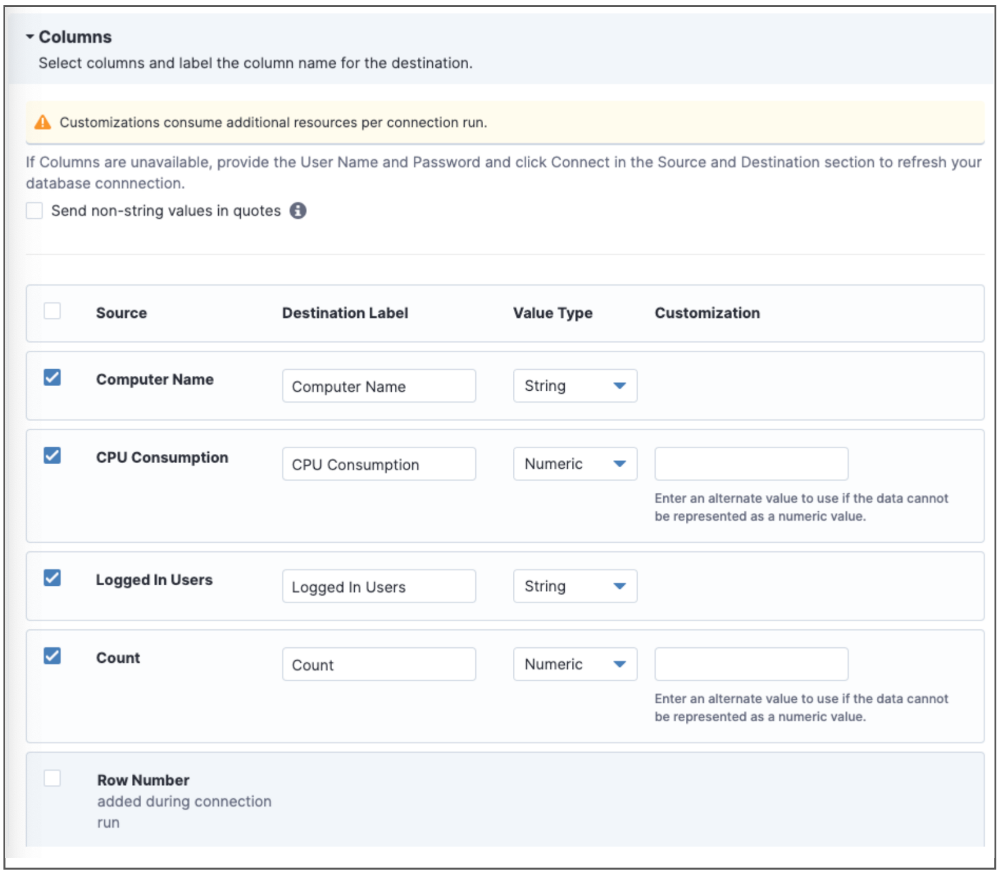
When you select a destination, the expected data format is displayed by default. For example, if you select Splunk, the `Syslog RFC 5424` automatically pre-populates the **Format Type** field. However, you can customize the format as needed. (For details, see Tanium's Format Types topic.)

Format Type: When sending to Cribl Stream, select the `JSON` data format for best results.

In the **Columns** section, configure the columns that you want to pass on to your Destination. (For details, see Tanium's Column Customizations topic.)

- **Source**: Check the box next to each **Source** to include the columns in your Destination.
- **Destination Labels**: You can optionally assign a new column heading. Defaults to the **Source** name.

- **Value Types**: You can change the data type to `String`, `Numeric`, or `Data/Time` value.
    - If you select a `Numeric` value, you must specify a default value. It can be any integer.
    - If you select a `Data/Time` value, specify the format to apply for the column. R(For details, see Tanium's Time Stamp Variables topic.)



Columns
Select columns and label the column name for the destination.

Customizations consume additional resources per connection run.

If Columns are unavailable, provide the User Name and Password and click Connect in the Source and Destination section to refresh your database connection.

Send non-string values in quotes

	Source	Destination Label	Value Type	Customization
✓	Computer Name	Computer Name	String	
✓	CPU Consumption	CPU Consumption	Numeric	Enter an alternate value to use if the data cannot be represented as a numeric value.
✓	Logged In Users	Logged In Users	String	
✓	Count	Count	Numeric	Enter an alternate value to use if the data cannot be represented as a numeric value.
	Row Number added during connection run			

Configure your columns

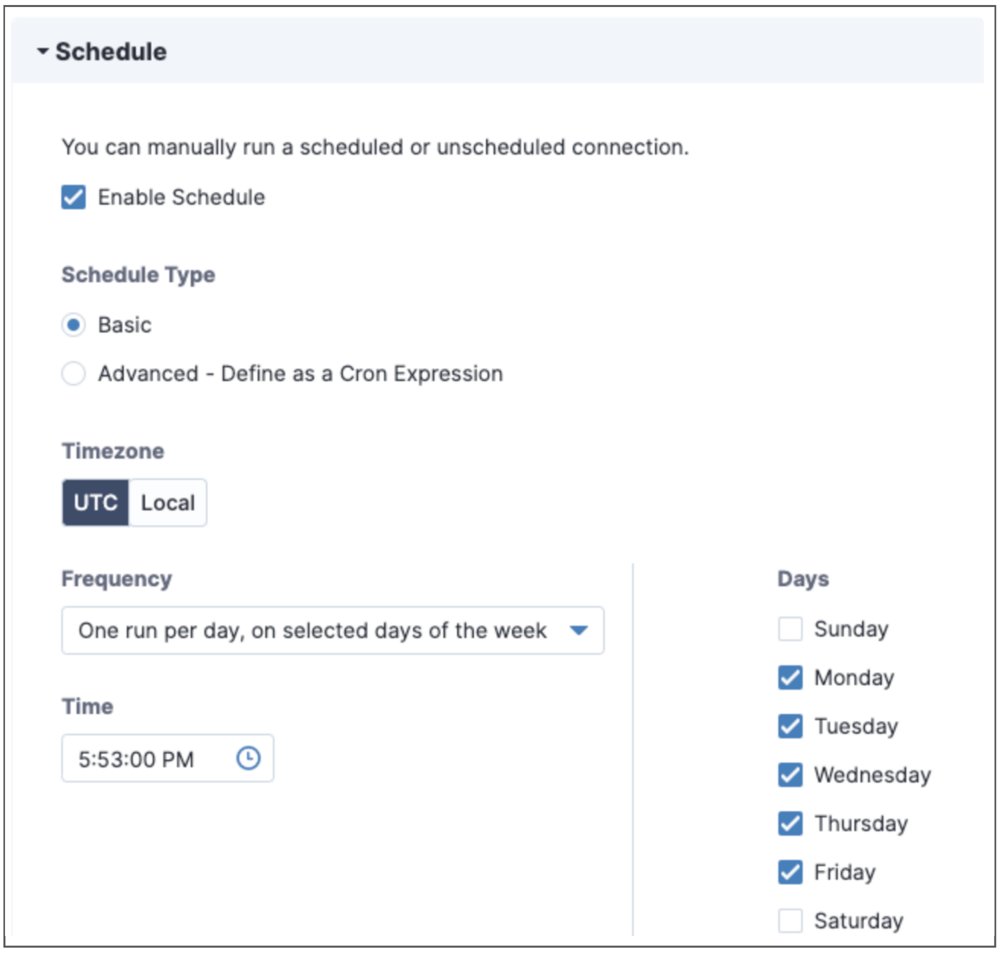
Schedule the Connection

Connections can run at a highly configurable time interval – anywhere from multiple times per hour, to daily, weekly, or monthly intervals. The **Schedule** section allows you to enable and configure the scheduler.

Enable Schedule: If you do not enable the scheduler here, the connection will run only when you explicitly run it.

Schedule Type: Select **Basic** to build a schedule with the provided controls.

Advanced – Define as a Cron Expression: Select this field to view or edit the cron expression directly.


▾ Schedule

You can manually run a scheduled or unscheduled connection.

☑ Enable Schedule

Schedule Type

● Basic

○ Advanced - Define as a Cron Expression

Timezone

UTC | Local

Frequency

One run per day, on selected days of the week

Time

5:53:00 PM

Days

☐ Sunday
☑ Monday
☑ Tuesday
☑ Wednesday
☑ Thursday
☑ Friday
☐ Saturday

Configure your columns

If the user who owns a connection is deactivated, future instances of a scheduled connection will not run. For details, see Tanium's Deleted User Troubleshooting topic.

Save and Verify the Connection

When you are done configuring your connection, click **Save**.

To view details when the connection runs, select the **Logs** tab. To inspect an individual run log, expand the row table.

For help on resolving errors, see Taniuum's Troubleshooting topic.

You can also click **Run and Save** to save and immediately run the connection. Connection details will be displayed for successful connections.

Configure Cribl Source

On your Cribl Stream instance, configure a TCP Source to receive the data from your configured Tanium connection. For a video demo of this step, see this Tanium blog post.

17.6.14. Configuring WEF For Cribl Stream

You can configure Cribl Stream to receive Windows events from the Windows Event Forwarding (WEF) mechanism. To do this, you configure a Windows endpoint/sender to forward events to Cribl Stream's Windows Event Forwarder Source with mutual TLS authentication.

This page picks up where the Windows Event Forwarder Source doc's upstream sender configuration section leaves off. Before you begin, make sure that you are running PowerShell as an administrator.

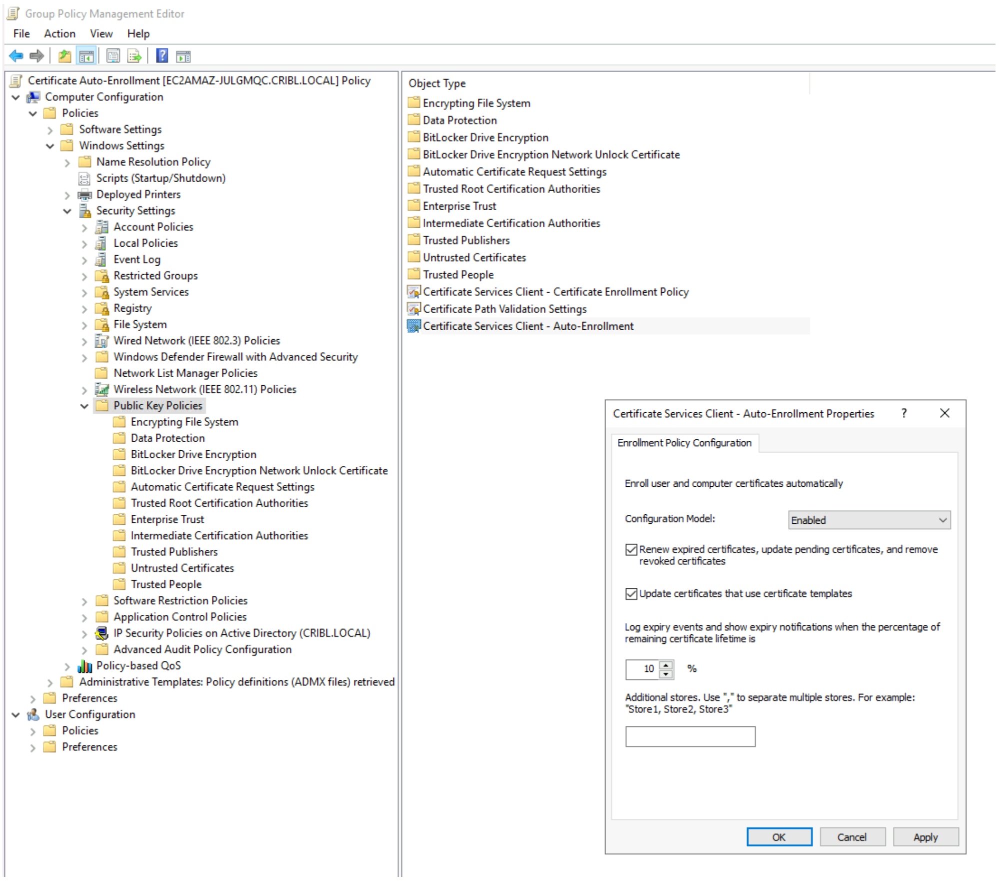
Configuring Certificate Auto-Enrollment

Cribl recommends using certificate auto-enrollment, via Group Policy, to configure unique device certificates for WEF. (However, if your environment is small enough that auto-enrollment seems like too heavyweight a procedure, skip ahead to Generating a CA and Client Certificates.)

In your Windows domain controller, edit or create a new Group Policy Object (GPO). Then, navigate to **Computer Configuration** > **Policies** > **Windows Settings** > **Security Settings** > **Public Key Policies**.

Double-click the **Certificate Services Client - Auto Enrollment Properties** Object Type. In the resulting modal:

- From the **Configuration Model** drop-down, select `Enabled`.
- Select the check box labeled:
  `Renew expired certificates, update pending certificates, and remove revoked certificates.`
- Select the check box labeled:
  `Update certificates that use certificate templates.`
- Click **OK** and **Apply**.


Group Policy Management Editor

File Action View Help

```
Certificate Auto-Enrollment [EC2AMAZ-JULGMQC.CRIBL.LOCAL] Policy
  Computer Configuration
    Policies
      Software Settings
      Windows Settings
        Name Resolution Policy
        Scripts (Startup/Shutdown)
        Deployed Printers
        Security Settings
          Account Policies
          Local Policies
          Event Log
          Restricted Groups
          System Services
          Registry
          File System
          Wired Network (IEEE 802.3) Policies
          Windows Defender Firewall with Advanced Security
          Network List Manager Policies
          Wireless Network (IEEE 802.11) Policies
          Public Key Policies
            Encrypting File System
            Data Protection
            BitLocker Drive Encryption
            BitLocker Drive Encryption Network Unlock Certificate
            Automatic Certificate Request Settings
            Trusted Root Certification Authorities
            Enterprise Trust
            Intermediate Certification Authorities
            Trusted Publishers
            Untrusted Certificates
            Trusted People
          Software Restriction Policies
          Application Control Policies
          IP Security Policies on Active Directory (CRIBL.LOCAL)
          Advanced Audit Policy Configuration
        Policy-based QoS
      Administrative Templates: Policy definitions (ADMX files) retrieved
    Preferences
  User Configuration
    Policies
    Preferences
```

Object Type
Encrypting File System
Data Protection
BitLocker Drive Encryption
BitLocker Drive Encryption Network Unlock Certificate
Automatic Certificate Request Settings
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Trusted People
Certificate Services Client - Certificate Enrollment Policy
Certificate Path Validation Settings
Certificate Services Client - Auto-Enrollment

Certificate Services Client - Auto-Enrollment Properties

Enrollment Policy Configuration

Enroll user and computer certificates automatically

Configuration Model: Enabled

[✓] Renew expired certificates, update pending certificates, and remove revoked certificates

[✓] Update certificates that use certificate templates

Log expiry events and show expiry notifications when the percentage of remaining certificate lifetime is

10 %

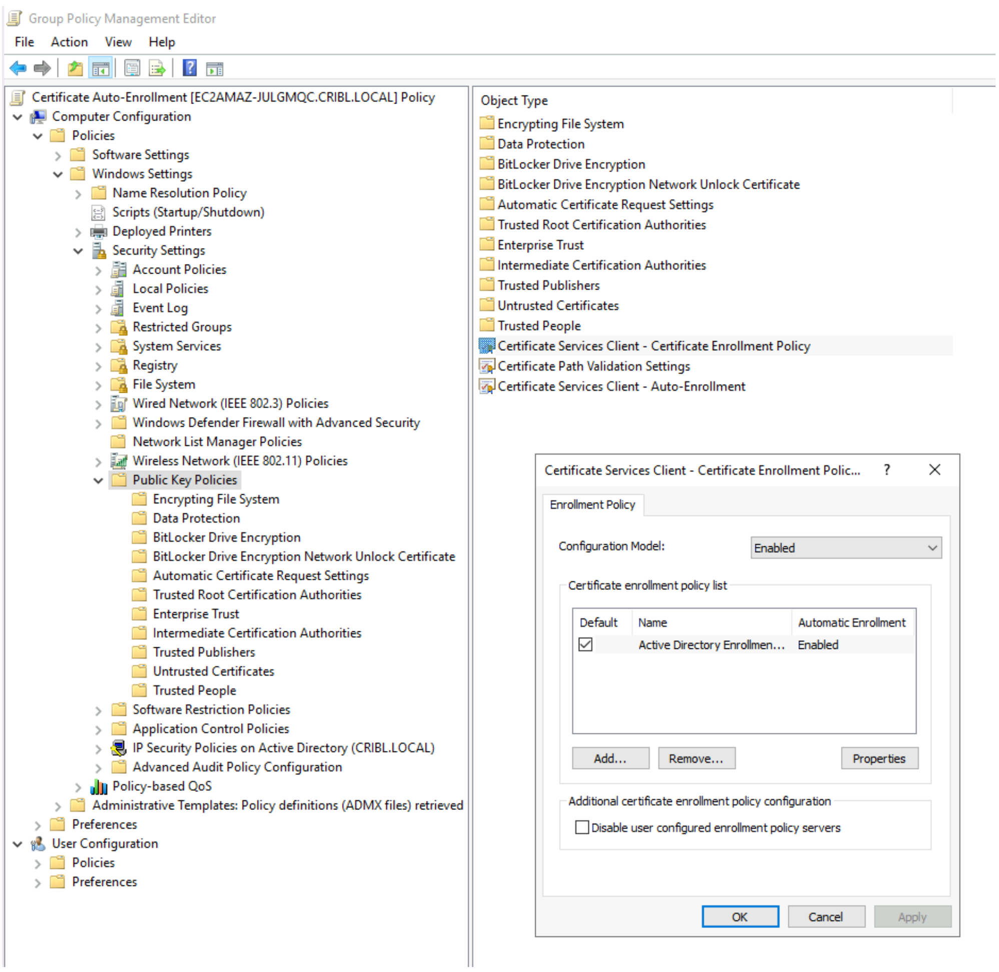
Additional stores. Use "," to separate multiple stores. For example: "Store1, Store2, Store3"

OK Cancel Apply

The Auto-Enrollment Properties modal

Select the **Certificate Services Client – Certificate Enrollment Policy** Object Type. In the resulting modal:

- From the **Configuration Model** drop-down, select `Enabled`.
- Click **OK** and **Apply**.



Group Policy Management Editor

File Action View Help

- Certificate Auto-Enrollment [EC2AMAZ-JULGMQC.CRIBL.LOCAL] Policy
  - Computer Configuration
    - Policies
      - Software Settings
      - Windows Settings
        - Name Resolution Policy
        - Scripts (Startup/Shutdown)
        - Deployed Printers
        - Security Settings
          - Account Policies
          - Local Policies
          - Event Log
          - Restricted Groups
          - System Services
          - Registry
          - File System
          - Wired Network (IEEE 802.3) Policies
          - Windows Defender Firewall with Advanced Security
          - Network List Manager Policies
          - Wireless Network (IEEE 802.11) Policies
          - Public Key Policies
            - Encrypting File System
            - Data Protection
            - BitLocker Drive Encryption
            - BitLocker Drive Encryption Network Unlock Certificate
            - Automatic Certificate Request Settings
            - Trusted Root Certification Authorities
            - Enterprise Trust
            - Intermediate Certification Authorities
            - Trusted Publishers
            - Untrusted Certificates
            - Trusted People
          - Software Restriction Policies
          - Application Control Policies
          - IP Security Policies on Active Directory (CRIBL.LOCAL)
          - Advanced Audit Policy Configuration
        - Policy-based QoS
      - Administrative Templates: Policy definitions (ADMX files) retrieved
    - Preferences
  - User Configuration
    - Policies
    - Preferences

Object Type
- Encrypting File System
- Data Protection
- BitLocker Drive Encryption
- BitLocker Drive Encryption Network Unlock Certificate
- Automatic Certificate Request Settings
- Trusted Root Certification Authorities
- Enterprise Trust
- Intermediate Certification Authorities
- Trusted Publishers
- Untrusted Certificates
- Trusted People
- Certificate Services Client - Certificate Enrollment Policy
- Certificate Path Validation Settings
- Certificate Services Client - Auto-Enrollment

Certificate Services Client - Certificate Enrollment Polic... ? ✕

Enrollment Policy

Configuration Model: Enabled

Certificate enrollment policy list

Default	Name	Automatic Enrollment
✓	Active Directory Enrollmen...	Enabled

Add... Remove... Properties

Additional certificate enrollment policy configuration

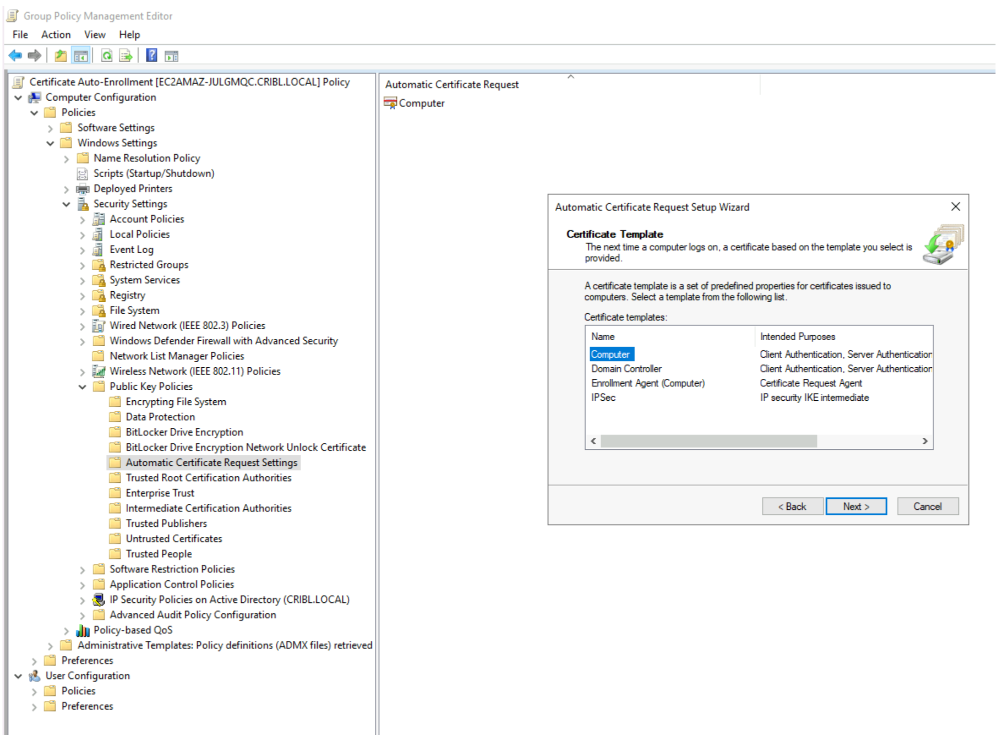
☐ Disable user configured enrollment policy servers

OK Cancel Apply

The Certificate Enrollment Policy modal

Double-click the **Automatic Certificate Request Settings** folder. In the resulting wizard:

- Select the **Computer** template, click **Next**, and follow the prompts to create a new automatic certificate request.


Group Policy Management Editor
File Action View Help
Certificate Auto-Enrollment [EC2AMAZ-JULGMQC.CRIBL.LOCAL] Policy
Computer Configuration
Policies
Software Settings
Windows Settings
Name Resolution Policy
Scripts (Startup/Shutdown)
Deployed Printers
Security Settings
Account Policies
Local Policies
Event Log
Restricted Groups
System Services
Registry
File System
Wired Network (IEEE 802.3) Policies
Windows Defender Firewall with Advanced Security
Network List Manager Policies
Wireless Network (IEEE 802.11) Policies
Public Key Policies
Encrypting File System
Data Protection
BitLocker Drive Encryption
BitLocker Drive Encryption Network Unlock Certificate
Automatic Certificate Request Settings
Trusted Root Certification Authorities
Enterprise Trust
Intermediate Certification Authorities
Trusted Publishers
Untrusted Certificates
Trusted People
Software Restriction Policies
Application Control Policies
IP Security Policies on Active Directory (CRIBL.LOCAL)
Advanced Audit Policy Configuration
Policy-based QoS
Administrative Templates: Policy definitions (ADMX files) retrieved
Preferences
User Configuration
Policies
Preferences
Automatic Certificate Request
Computer
Automatic Certificate Request Setup Wizard
< Back
Next >
Cancel

Certificate Template

The next time a computer logs on, a certificate based on the template you select is provided.

A certificate template is a set of predefined properties for certificates issued to computers. Select a template from the following list.

Certificate templates:

Name	Intended Purposes
Computer	Client Authentication, Server Authenticatior
Domain Controller	Client Authentication, Server Authenticatior
Enrollment Agent (Computer)	Certificate Request Agent
IPSec	IP security IKE intermediate

The Automatic Certificate Request Setup Wizard

Generating a CA and Client Certificates

If you followed the preceding Configuring Certificate Auto-Enrollment section to configure certificates, skip ahead to Setting Client Certificate Permissions.

To begin, generate a public/private key pair that constitute a new Cribl WEF Certificate Authority (CA) – i.e., the root CA. Then, import it into the local computer's Trusted Root CA store. The following PowerShell commands do all of this (run as Administrator):

```
$rootca = New-SelfSignedCertificate -CertStoreLocation cert:\LocalMachine\My -DnsNa
Export-Certificate -Cert $rootca -FilePath C:\temp\cribl-wef-ca.cer
Import-Certificate -FilePath C:\temp\cribl-wef-ca.cer -CertStoreLocation cert:\Loca
```

Now, generate the client certificate.

```
New-SelfSignedCertificate -CertStoreLocation cert:\LocalMachine\My -DnsName cribl-w
```

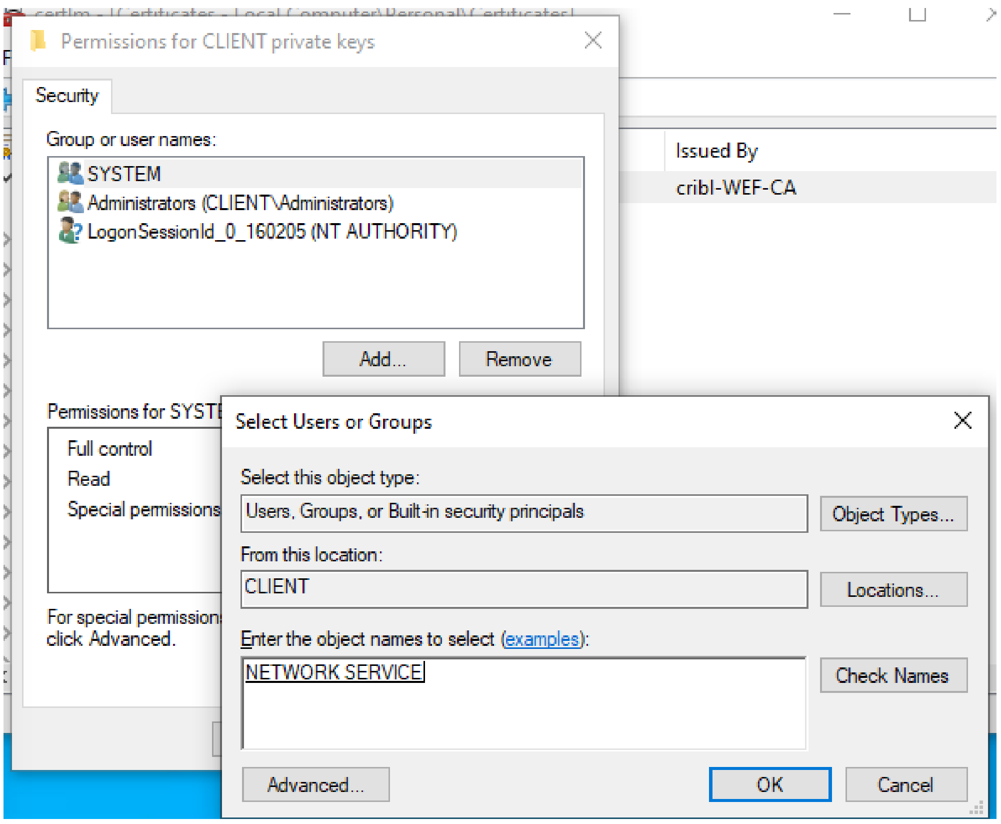
Setting Client Certificate Permissions

You can perform this step through the Windows UI, or on the command line.

Cert Permissions via UI

Open the Certificate Manager tool (`certlm.msc`).

- Right-click the `cribl-wef-client` certificate and select **All Tasks** > **Manage Private Keys...**.
- Add the `NETWORK SERVICE` user.



Adding the Network Service user

Cert Permissions via Command Line

As an alternative, you can run the following PowerShell script (as administrator):

```
$issuer = "CN=CRIBL-CA, DC=cribl, DC=local"
$certs = Get-ChildItem -Path cert:\LocalMachine\My | Where {$_.Issuer -eq $issuer}

Foreach ($cert in $certs)
{
  # Specify the user, the permissions, and the permission type
  $permission = "Network Service","FullControl","Allow"
  $accessRule = New-Object -TypeName System.Security.AccessControl.FileSystemAcces

  # Location of the machine-related keys
  $keyPath = Join-Path -Path $env:ProgramData -ChildPath "\Microsoft\Crypto\RSA\Ma
  $keyName = $cert.PrivateKey.CspKeyContainerInfo.UniqueKeyContainerName
  $keyFullPath = Join-Path -Path $keyPath -ChildPath $keyName

  # Get the current ACL (Access Control List) of the private key
  $acl = (Get-Item $keyFullPath).GetAccessControl('Access')

  # Add the new ACE to the ACL of the private key
  # An ACE (Access Control Entry) is an individual rule in an ACL
  $acl.SetAccessRule($accessRule)

  # Write back the new ACL
  Set-Acl -Path $keyFullPath -AclObject $acl -ErrorAction Stop

  # Observe the access rights currently assigned to this certificate.
  get-acl $keyFullPath| fl
}
```

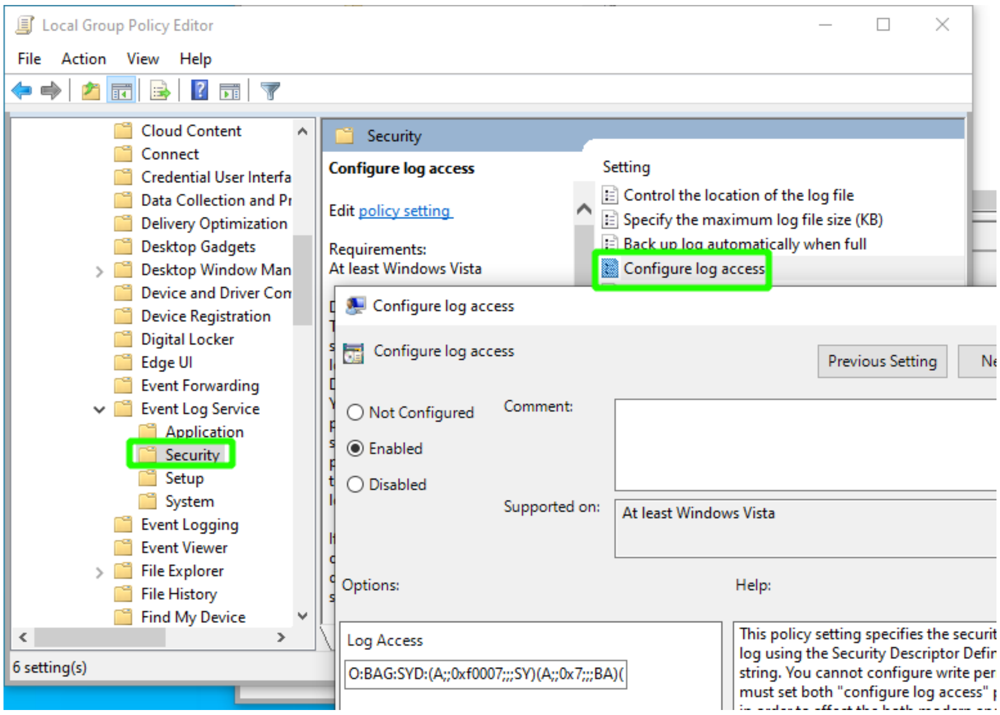
(Source: Stack Overflow – NETWORK SERVICE user cannot read certificate Windows Server 2012.)

Configuring Log Forwarding Group Policies

Your goal here is to enable the Network Service to read the security logs. To do this, you'll configure access for the Event Log Service.

Open the Local Group Policy Editor tool (`gpedit.msc`).

Navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components** > **Event Log Service**. Double-click **Security**, then in the **Settings** pane, select **Configure log access**.



Not Configured [✓ Enabled] Disabled

Log Access: O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(

This policy setting specifies the securit log using the Security Descriptor Defir string. You cannot configure write per must set both "configure log access"

Configuring log access

In the resulting modal, under **Options** > **Log Access**, enter the following Log Access configuration:

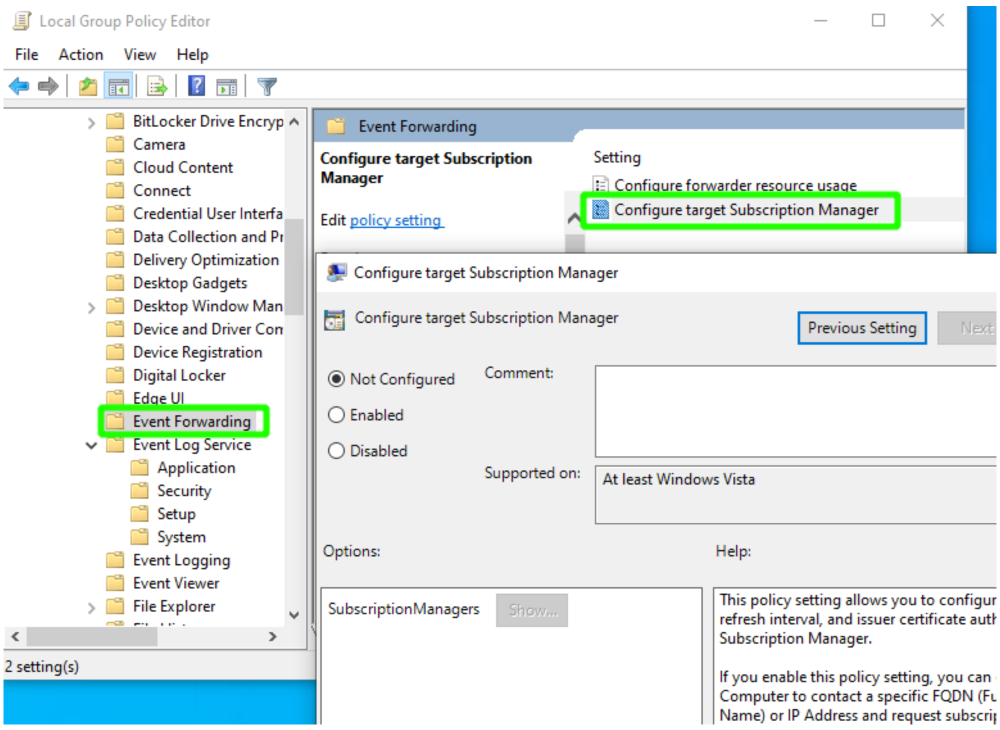
```
O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)(A;;0x1;;;SO)(A;;0x1;;;S-1-5-
```

Then, reboot your Windows machine to apply this setting.

(Source: Microsoft’s Security event log forwarding fails… topic.)

Configuring the Subscription Manager

Navigate to **Computer Configuration** > **Policies** > **Administrative Templates** > **Windows Components**. Select **Event Forwarding** to open the Local Group Policy Editor.



Configure target Subscription Manager

Configure target Subscription Manager

(● Not Configured) ○ Enabled ○ Disabled

Comment:

Supported on: At least Windows Vista

Options:

SubscriptionManagers

Help:

This policy setting allows you to configur
refresh interval, and issuer certificate auth
Subscription Manager.

If you enable this policy setting, you can
Computer to contact a specific FQDN (Fu
Name) or IP Address and request subscrip

Configuring the Subscription Manager

Select **Configure target Subscription Manager** and enter the following template string, substituting the appropriate values for the placeholders:

```
Server=https://in.main-default-<organization>.cribl.cloud:<port>/wsman/Subscriptio
```

If you forget your Client CA hash, run the following command to find it:

```
Get-ChildItem cert:\LocalMachine\root | Where-Object {$_.Subject -Like "CN=Cr
```

Configuring the CA Certificate in Cribl.Cloud

Complete this section only if your deployment is in Cribl.Cloud. If your Cribl Stream deployment is on-prem or hybrid, import the correct certificate for the appliance, then skip ahead to Configuring a New WEF Source.

First, dump the CA certificate in PEM format:

```
$oMachineCert=get-item cert:\LocalMachine\Root\$($(Get-ChildItem cert:\LocalMachine
$oPem=new-object System.Text.StringBuilder
$oPem.AppendLine("-----BEGIN CERTIFICATE-----")
$oPem.AppendLine([System.Convert]::ToBase64String($oMachineCert.RawData,1))
$oPem.AppendLine("-----END CERTIFICATE-----")
$oPem.ToString() | out-file C:\cribl-wef-ca.pem
```

The `cribl-wef-ca` certificate should now be available at `C:\cribl-wef-ca.pem`.

Cribl Stream requires every CA certificate to be accompanied by a cert/key pair. Although Cribl Stream won't ever use the cert/key pair for the `cribl-wef-ca` certificate, you need to generate this pair to pass validation.

Run the following `openssl` command to generate a placeholder cert/key pair:

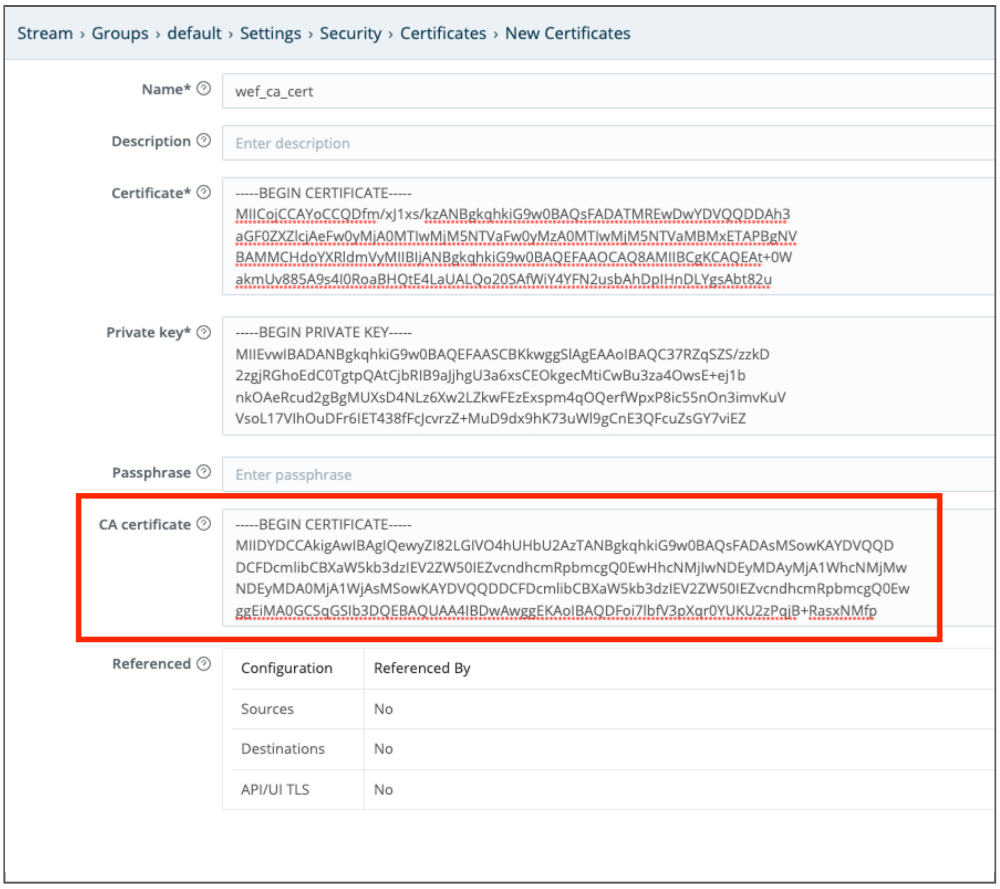
```
openssl req -x509 -newkey rsa:2048 -nodes -keyout key.pem -out cert.pem -sha256 -da
cat cert.pem key.pem
```

In the Cribl Stream Worker Group UI, navigate to **Settings** > **Security** > **Certificates** > **New Certificates**.

Paste the placeholder certificate and key into the **Certificate** and **Private key** fields, respectively. Cribl Stream automatically pastes the `cribl-wef-ca` PEM contents into the **CA certificate** field.

If the client certificates contain a CA chain (root and intermediate signers), you must import the entire CA chain. Concatenate the PEM files together in the **CA certificate** field.

Each certificate here must directly certify the one preceding it. This matches the order of traversing the chain from the host to the root CA cert.


Stream › Groups › default › Settings › Security › Certificates › New Certificates

Name*	wef_ca_cert
Description	Enter description
Passphrase	Enter passphrase

Certificate*

```
-----BEGIN CERTIFICATE-----
MIICojCCAYoCCQDfm/xJ1xs/kzANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAh3
aGF0ZXZlcjAeFw0yMjA0MTIwMjM5NTVaFw0yMzA0MTIwMjM5NTVaMBMxETAPBgNV
BAMMCHdoYXRldmVyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+0W
akmUv885A9s4I0RoaBHQtE4LaUALQo20SAfWiY4YFN2usbAhDpIHnDLYgsAbt82u
```

Private key*

```
-----BEGIN PRIVATE KEY-----
MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQC37RZqSZS/zzkD
2zgjRGhoEdC0TgtpQAtCjbRIB9aJjhgU3a6xsCEOkgecMtiCwBu3za4OwsE+ej1b
nkOAeRcud2gBgMUXsD4NLz6Xw2LZkwFEzExspm4qOQerfWpxP8ic55nOn3imvKuV
VsoL17VIhOuDFr6IET438fFcJcvrzZ+MuD9dx9hK73uWl9gCnE3QFcuZsGY7viEZ
```

CA certificate

```
-----BEGIN CERTIFICATE-----
MIIDYDCCAkigAwIBAgIQewyZI82LGIVO4hUHbU2AzTANBgkqhkiG9w0BAQsFADAsMSowKAYDVQQD
DCFDcmlibCBXaW5kb3dzIEV2ZW50IEZvcndhcmRpbmcgQ0EwHhcNMjIwNDEyMDAyMjA1WhcNMjMw
NDEyMDA0MjA1WjAsMSowKAYDVQQDDCFDcmlibCBXaW5kb3dzIEV2ZW50IEZvcndhcmRpbmcgQ0Ew
ggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDFoi7lbfV3pXqr0YUKU2zPqjB+RasxNMfp
```

Referenced

Configuration	Referenced By
Sources	No
Destinations	No
API/UI TLS	No

Configuring certificates

When the WEF machine connects to Cribl Stream, the machine presents its certificate to Cribl Stream. To authenticate clients that try to send logs from WEF, Cribl Stream compares the signer of the WEF machine's certificate against the CA list (which you imported in the previous step).

Configuring a New WEF Source

To create a new Windows Event Forwarder Source:

In the **QuickConnect** UI: Click **Add Source**. From the resulting drawer's tiles, select [**Push** >] **Windows Event Forwarder**. Next, click either **Add Destination** or (if displayed) **Select Existing** to open a Windows Event Forwarder Source drawer.

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 [**Push** >] **Windows Event Forwarder**. Next, click **New Source** to open a **New Source** modal.

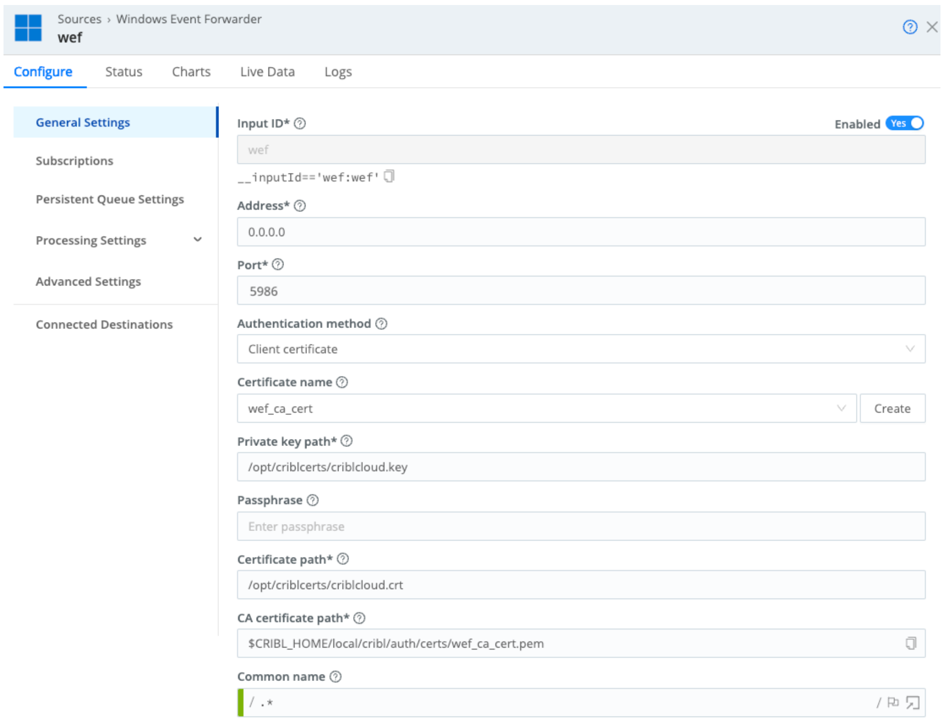
The drawer or modal will now provide the options and fields described in the Windows Event Forwarder Source topic. That topic describes the general case; this one covers only settings that are specific to the use case we're describing.

General Settings

In the **Port** field, do not change the default of `5986` – that is the port WEF uses for mutual TLS authentication.

From the **Certificate name** drop-down, choose the certificate you just created:

- If you are using a hybrid Worker Group, provide the path to your own key and certificate in **Private key path** and **Certificate path**.
- If you are using a Cribl-managed Cribl.Cloud Worker Group, replace the contents of **Private key path** with `/opt/criblcerts/criblcloud.key`, and the contents of **Certificate path** with `/opt/criblcerts/criblcloud.crt`. (These are the default certificates that ship with Cribl.Cloud.)



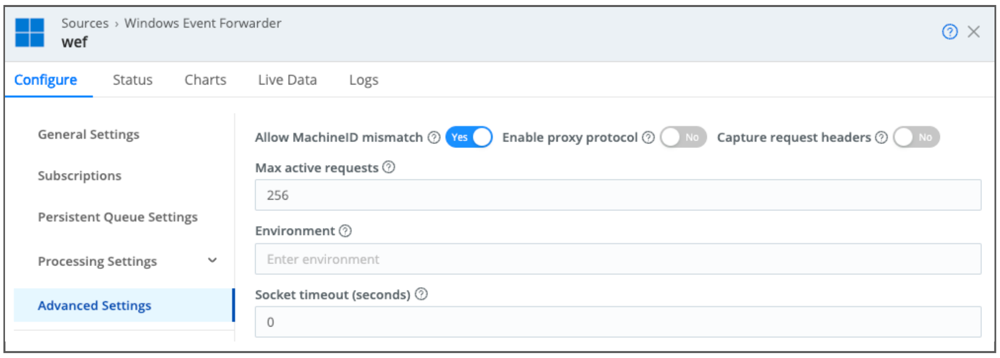
Input ID*: wef
Enabled: Yes
`__inputId=='wef:wef'`
Address*: 0.0.0.0
Port*: 5986
Authentication method: Client certificate
Certificate name: wef_ca_cert
Private key path*: /opt/criblcerts/criblcloud.key
Passphrase: Enter passphrase
Certificate path*: /opt/criblcerts/criblcloud.crt
CA certificate path*: $CRIBL_HOME/local/cribl/auth/certs/wef_ca_cert.pem
Common name: / .*

Configuring General Settings

Advanced Settings

If you're using a shared certificate across multiple machines (meaning that your devices cannot be configured for certificate auto-enrollment), toggle **Allow MachineID mismatch** to `Yes`. This tells Cribl Stream to allow machine names that do not match the common name (CN) specified in the certificate. However, certificates will still need to be signed by the Certificate Authority that you uploaded.

If your devices **are** configured for certificate auto-enrollment, toggle **Allow MachineID mismatch** to `No`, because `MachineIDs` will match the `Subject Name` specified the certificate. The `No` setting provides better security than `Yes`.


Sources › Windows Event Forwarder
wef

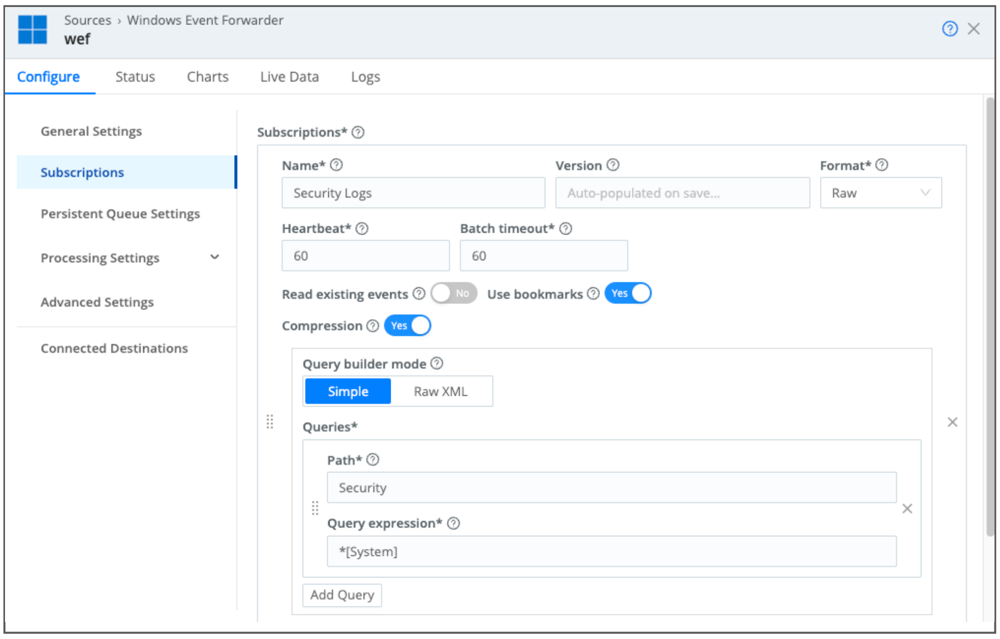
Allow MachineID mismatch: Yes
Enable proxy protocol: No
Capture request headers: No
Max active requests: 256
Environment: Enter environment
Socket timeout (seconds): 0

Configuring Advanced Settings

Subscriptions

Configure a subscription to forward the type of logs you're interested in. For example, to forward all Security logs:

- In **Query** > **Path**, enter `Security`.
- In **Query** > **Query expression**, enter `*[System]`.


Sources › Windows Event Forwarder
wef

Subscriptions*
Name*: Security Logs
Version: Auto-populated on save...
Format*: Raw
Heartbeat*: 60
Batch timeout*: 60
Read existing events: No
Use bookmarks: Yes
Compression: Yes
Query builder mode: Simple
Queries*
Path*: Security
Query expression*: *[System]

Configuring Subscriptions

Remember to save, commit, and deploy your new configuration **as soon as you have configured Subscriptions**. This is required for the previous steps, beginning with Configuring the CA Certificate in Cribl.Cloud, to take effect.

When you first create a subscription, Windows Event Forwarder generates its subscription ID. Every `Refresh=XX` seconds – as defined in the Group Policy Objects (GPO) – the WEF clients contact Cribl Stream (or Cribl Edge) to ask for a subscription. Upon receiving the subscription, the clients start sending events, referencing the subscription ID. The clients send events at the interval specified in the WEF Source **Batch timeout** setting.

Changing Subscriptions

If you change a subscription that is in use, Windows Event Forwarder will discover the new subscription ID. Once the `Refresh=XX` interval has elapsed, if any events were sent to the now-invalid previous subscription ID, Windows will resend them to the new one (assuming that these events remain valid under the updated subscription parameters). The Receiver Refuses to Accept Delivery of Events warnings you'll see in this scenario are normal.

Troubleshooting

A full account of how to troubleshoot WEF certificate administration is beyond the scope of this guide, but knowing about the following common errors is a good start.

Cannot find the Certificate

Error:
`The forwarder is having a problem communicating with subscription manager at address… The WS-Management service cannot find the certificate that was requested.`

For example:

```
The forwarder is having a problem communicating with subscription manager at addres

Error code is 2150858882 and Error Message is <f:WSManFault xmlns:f=http://schemas
```

Resolution:

- The `NETWORK SERVICE` user does not have permissions to use the private key of the certificate for authentication.
- Add the `NETWORK SERVICE` user to the list of users allowed to use the private key.

Receiver Refuses to Accept Delivery of Events

Warning:
`The receiver refuses to accept delivery of events and requests that the subscription be canceled.`

If you make a change to a subscription that is in use, **and** if the Windows client tries to send events before the `Refresh=XX` interval has elapsed, the result is that these events are sent to the old subscription, which is now invalid because its query is outdated or because other subscription parameters have changed.

The warning message tells the WEF client to stop sending events to the now-invalid old subscription.

Resolution:

Windows Event Forwarder handles this scenario gracefully, and you do not need to take any action to resolve it:

- Once the `Refresh=XX` interval has elapsed, the WEF client will ask for and receive the updated subscription, and from then on should be able to successfully deliver events against that new subscription.
- Any events that were refused under the old/invalid subscription ID will be resent under the new one, assuming that they remain valid according to the new subscription parameters.

Subscription Cannot be Created

Error:
`The subscription <subscription> can not be created. The error code is 5004.`

The error appears in the `Microsoft-Windows-Eventlog-ForwardingPlugin/Operational` log.

Resolution:

1. Apply the `O:BAG:SYD:...` permissions to make the log file readable.
2. Reboot the machine.

Verify that Client and Destination are Joined to a Domain

Error:
`If Kerberos mechanism is used, verify that the client computer and the destination computer are joined to a domain.`

When you are using auto-enroll, the `Eventlog-ForwardingPlugin Operational` logs might display an error of this form.

Resolution:

In your auto-enroll template’s **Client-Server Authentication Properties**, make sure the **Subject Name format** is set to **Common name**.

Further Reading

For more context and greater detail, see:

- System Center Dudes' blog post about using PowerShell to generate certs.
- ATA Learning's WEF tutorial.

For more about WEF subscriptions, including sample subscription queries, see:

- Microsoft's Use Windows Event Forwarding to Help with Intrusion Detection topic.
- The NSA Cybersecurity Directorate's WEF Guidance.
- Palantir's WEF Guidance.

17.6.15. Zscaler NSS Integration

Zscaler Nanolog Streaming Service (NSS) uses a VM to stream traffic logs in real time from the Zscaler Nanolog to a SIEM. Cribl Stream can take the place of the SIEM in this arrangement. Then you can use Cribl Stream to greatly reduce the size of ZScaler logs.

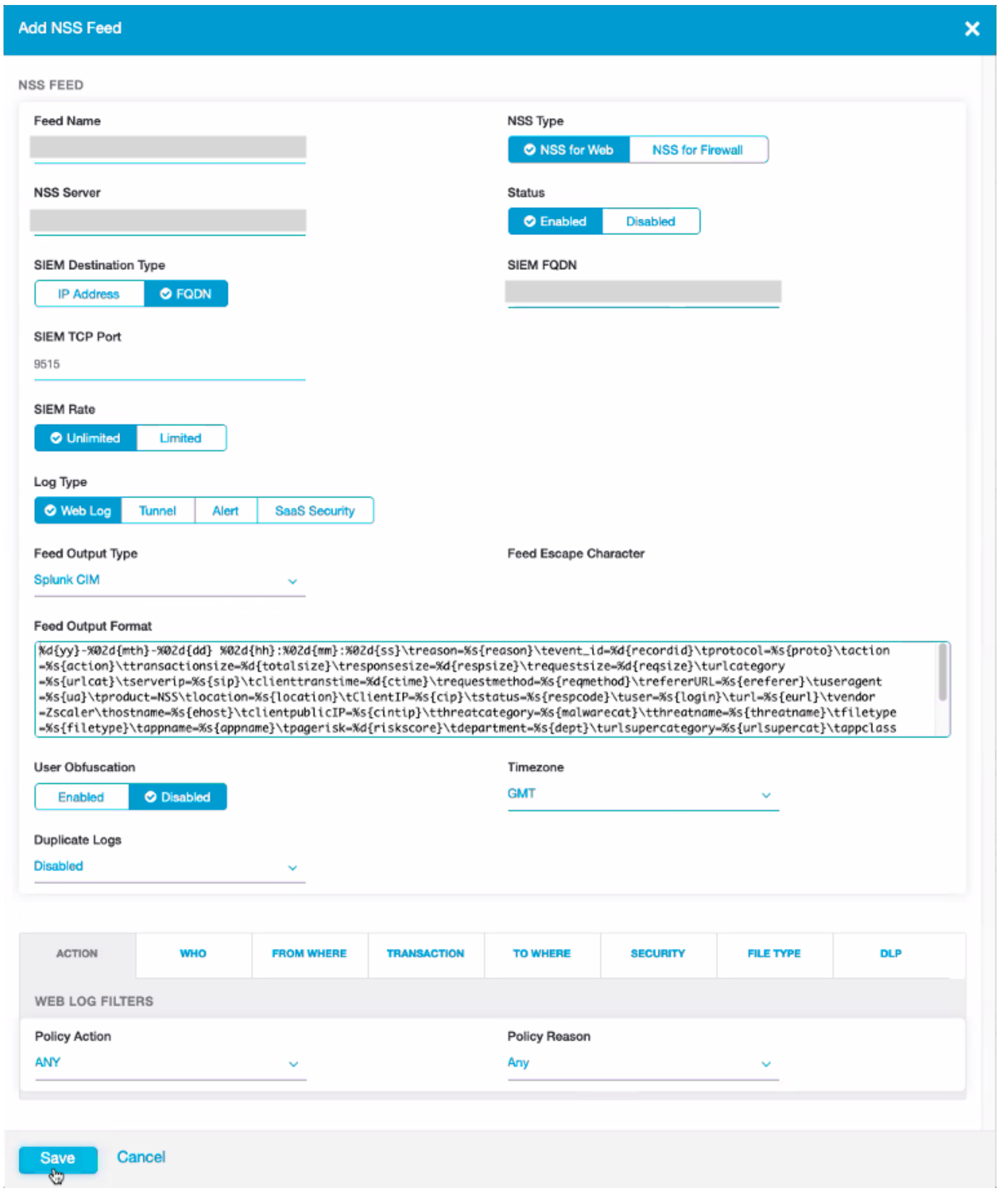
Configuring NSS to Send Data to Cribl Stream

Since Nanolog forwards data to a single IP address or FQDN, Cribl recommends that you use a load balancer to distribute data among Cribl Stream Workers.

Nanolog delivers data using a raw TCP connection.

In Zscaler:

- Go to **Administration** > **Nanolog Streaming Service**.
- In the **NSS Feeds** tab, click **Add NSS Feed** to open the following configuration window:


Add NSS Feed

NSS FEED

Feed Name: [redacted]

NSS Type: [✓ NSS for Web] NSS for Firewall

NSS Server: [redacted]

Status: [✓ Enabled] Disabled

SIEM Destination Type: IP Address [✓ FQDN]

SIEM FQDN: [redacted]

SIEM TCP Port: 9515

SIEM Rate: [✓ Unlimited] Limited

Log Type: [✓ Web Log] Tunnel Alert SaaS Security

Feed Output Type: Splunk CIM

Feed Escape Character:

Feed Output Format:

```
%d{yy}-%02d{mth}-%02d{dd} %02d{hh}:%02d{mm}:%02d{ss}\treason=%s{reason}\tevent_id=%d{recordid}\tprotocol=%s{proto}\taction
=%s{action}\ttransactionsize=%d{totalsize}\tresponsesize=%d{respsize}\trequestsize=%d{reqsize}\turlcategory
=%s{urlcat}\tserverip=%s{sip}\tclienttranstime=%d{ctime}\trequestmethod=%s{reqmethod}\trefererURL=%s{ereferer}\tuseragent
=%s{ua}\tproduct=NSS\tlocation=%s{location}\tClientIP=%s{cip}\tstatus=%s{respcode}\tuser=%s{login}\turl=%s{eurl}\tvendor
=Zscaler\thostname=%s{ehost}\tclientpublicIP=%s{cintip}\tthreatcategory=%s{malwarecat}\tthreatname=%s{threatname}\tfiletype
=%s{filetype}\tappname=%s{appname}\tpagerisk=%d{riskscore}\tdepartment=%s{dept}\turlsupercategory=%s{urlsupercat}\tappclass
```

User Obfuscation: Enabled [✓ Disabled]

Timezone: GMT

Duplicate Logs: Disabled

WEB LOG FILTERS

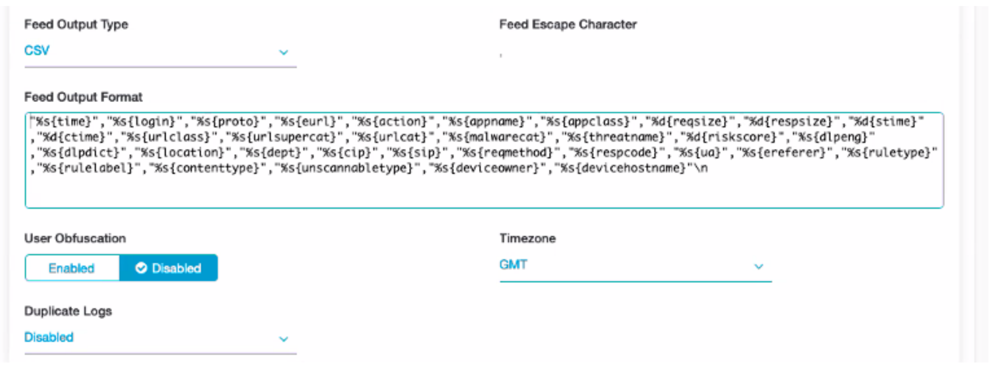
Policy Action: ANY

Policy Reason: Any

Adding a Zscaler NSS feed

- Enter a **Feed Name** that identifies this feed as one that sends data to Cribl Stream.
- Enter the IP address or FQDN for either your Cribl Stream instance, or the load balancer you're using with your Cribl Stream instances.
- Select a **Feed Output Type**. `Splunk CIM`, a tab-delimited key/value format, is a typical choice.

Alternatively, you choose a different option, such as CSV:



Feed Output Type: CSV

Feed Escape Character: ,

Feed Output Format:

```
"%s{time}","%s{login}","%s{proto}","%s{eurl}","%s{action}","%s{appname}","%s{appclass}","%d{reqsize}","%d{respsize}","%d{stime}"
,"%d{ctime}","%s{urlclass}","%s{urlsupercat}","%s{urlcat}","%s{malwarecat}","%s{threatname}","%d{riskscore}","%s{dlpeng}"
,"%s{dlpdict}","%s{location}","%s{dept}","%s{cip}","%s{sip}","%s{reqmethod}","%s{respcode}","%s{ua}","%s{ereferer}","%s{ruletype}"
,"%s{rulelabel}","%s{contenttype}","%s{unscannabletype}","%s{deviceowner}","%s{devicehostname}"\n
```

User Obfuscation: Enabled [✓ Disabled]

Timezone: GMT

Duplicate Logs: Disabled

Choosing a CSV output format

Example Pipeline

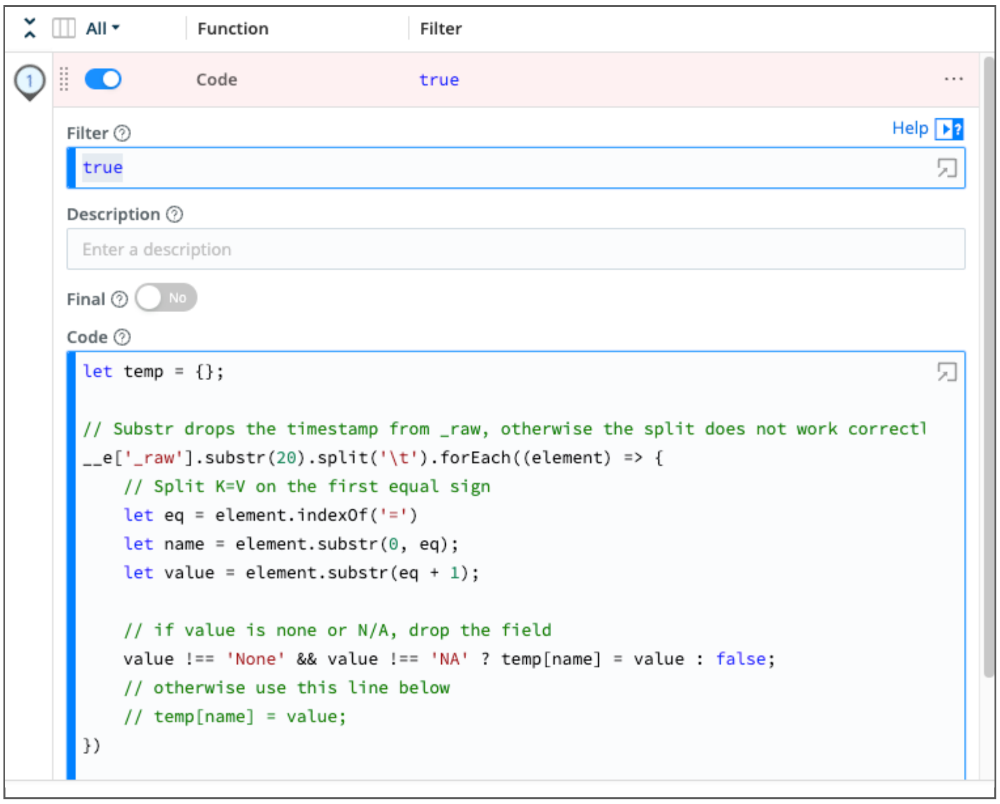
Cribl Stream can reduce Zscaler log size by (1) reformatting and reshaping the data, and (2) suppressing, sampling, and dropping appropriate fields.

The following code block shows how to correctly parse tab-delimited key-value pairs.

```
let temp = {};

// Substr drops the timestamp from _raw, otherwise the split does not work correctl
__e['_raw'].substr(20).split('\t').forEach((element) => {
    // Split K=V on the first equal sign
    let eq = element.indexOf('=')
    let name = element.substr(0, eq);
    let value = element.substr(eq + 1);

    // if value is none or N/A, drop the field
    value !== 'None' && value !== 'NA' ? temp[name] = value : false;
    // otherwise use this line below
    // temp[name] = value;
})

__e['_raw'] = temp;
```

Here's an example Pipeline that uses the parsing code above. (You can directly import this Pipeline in JSON form.)

A Code Function parses the data:



All	Function	Filter
1	Code	true

Filter
true

Description
Enter a description

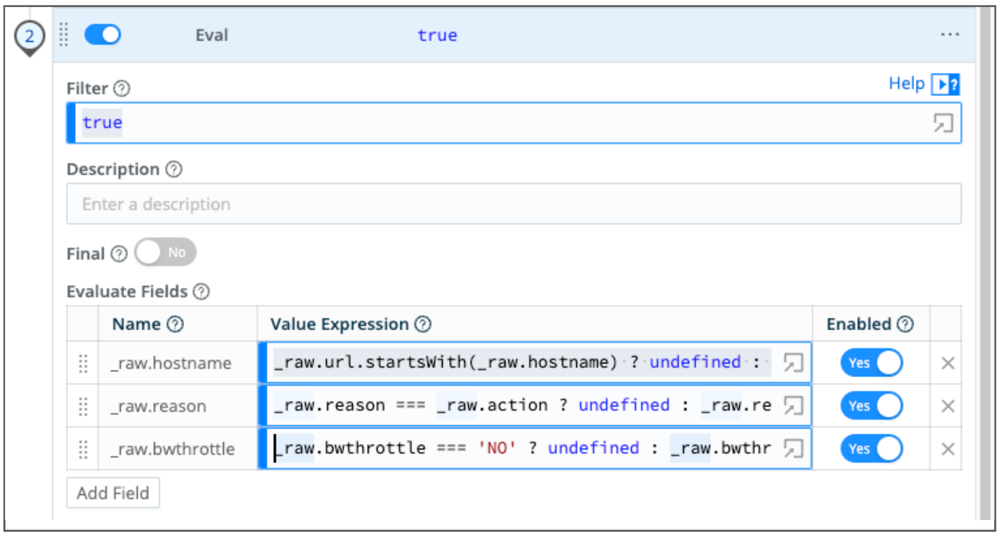
Final No

Code

```
let temp = {};

// Substr drops the timestamp from _raw, otherwise the split does not work correctl
__e['_raw'].substr(20).split('\t').forEach((element) => {
    // Split K=V on the first equal sign
    let eq = element.indexOf('=')
    let name = element.substr(0, eq);
    let value = element.substr(eq + 1);

    // if value is none or N/A, drop the field
    value !== 'None' && value !== 'NA' ? temp[name] = value : false;
    // otherwise use this line below
    // temp[name] = value;
})
```

Parsing with a Code Function

An Eval Function reshapes the data:



2 Eval true

Filter
true

Description
Enter a description

Final No

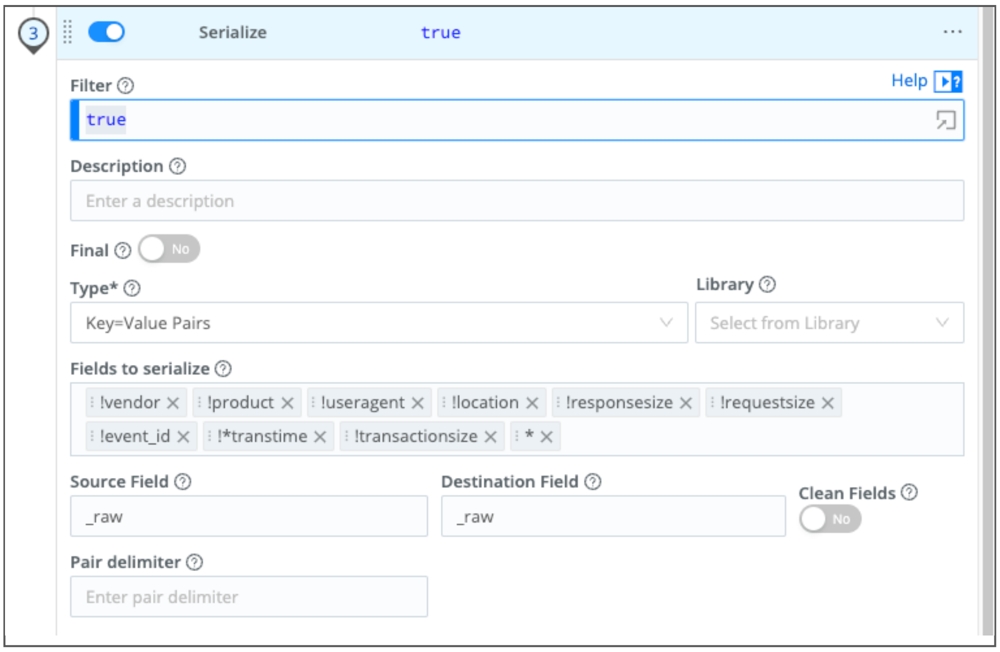
Evaluate Fields

Name	Value Expression	Enabled
_raw.hostname	`_raw.url.startsWith(_raw.hostname) ? undefined :`	Yes
_raw.reason	`_raw.reason === _raw.action ? undefined : _raw.re`	Yes
_raw.bwthrottle	`_raw.bwthrottle === 'NO' ? undefined : _raw.bwthr`	Yes

Add Field

Reshaping with Eval

And finally, a Serialize Function drops unwanted fields:



Serialize true

Filter: true

Description: Enter a description

Final: No

Type*: Key=Value Pairs

Library: Select from Library

Fields to serialize: !vendor, !product, !useragent, !location, !responsesize, !requestsize, !event_id, !*transtime, !transactionsize, *

Source Field: _raw

Destination Field: _raw

Clean Fields: No

Pair delimiter: Enter pair delimiter

Dropping fields with Serialize

Example Pipeline JSON

To import the example Pipeline directly, copy and save the JSON below, then follow these instructions.

Zscaler Example Pipeline

```
{
  "id": "zscaler",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "id": "code",
        "filter": "true",
        "disabled": false,
        "conf": {
          "maxNumOfIterations": 5000,
          "code": "let temp = {};\n\n// Substr drops the timestamp from _raw, other
        }
      },
      {
        "id": "eval",
        "filter": "true",
        "disabled": false,
        "conf": {
          "add": [
            {
              "name": "_raw.hostname",
              "value": "_raw.url.startsWith(_raw.hostname) ? undefined : _raw.hostn
            },
            {
              "name": "_raw.reason",
              "value": "_raw.reason === _raw.action ? undefined : _raw.reason"
            },
            {
              "name": "_raw.bwthrottle",
              "value": "_raw.bwthrottle === 'NO' ? undefined : _raw.bwthrottle"
            }
          ]
        }
      },
      {
        "id": "serialize",
        "filter": "true",
        "disabled": false,
        "conf": {
          "type": "kvp",
          "fields": [
            "!vendor",
            "!product",
            "!useragent",
            "!location",
            "!responsesize",
            "!requestsize",
            "!event_id",
            "!*transtime",
            "!transactionsize",
            "*"
          ],
          "dstField": "_raw",
          "cleanFields": false,
          "srcField": "_raw"
        }
      }
    ]
  }
}
```

18. Knowledge Libraries

Select **Processing** > **Knowledge** to access Cribl Stream'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:

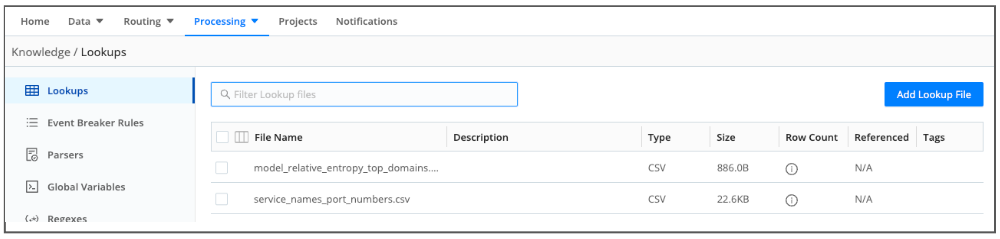
- Lookups
- Event Breakers
- Parsers
- Global Variables
- Regexes
- Grok Patterns
- Schemas (JSON)
- Parquet Schemas
- Database Connections (currently supported only in Cribl Stream)
- AppScope Configs

18.1. Lookups Library

What Are Lookups

Lookups are data tables that can be used in Cribl Stream 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 Stream'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 Stream's UI.



File Name	Description	Type	Size	Row Count	Referenced	Tags
model_relative_entropy_top_domains....		CSV	886.0B	ⓘ	N/A	
service_names_port_numbers.csv		CSV	22.6KB	ⓘ	N/A	

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 Stream's default Git integration. For details, see Managing Large Lookups.

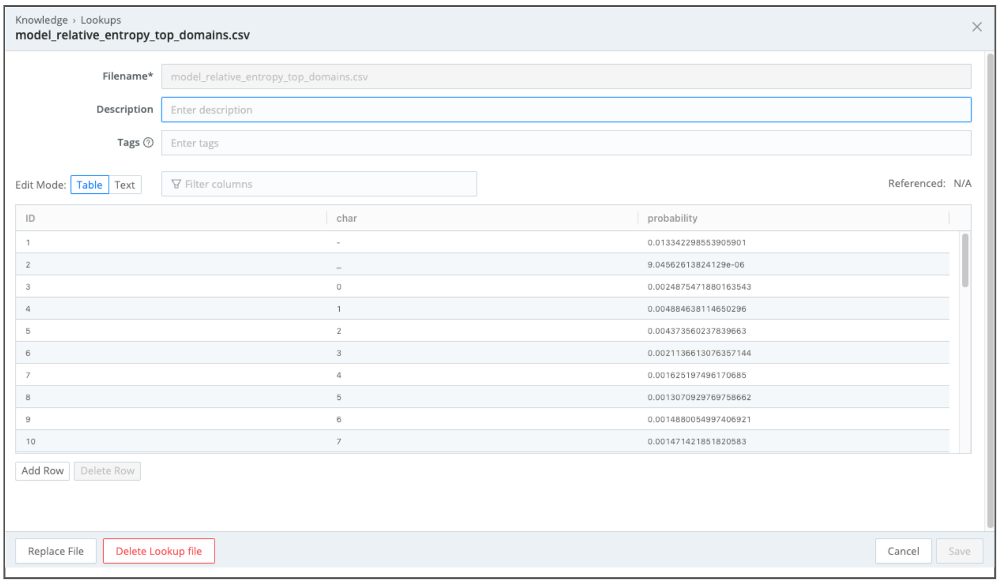
Adding Lookup Files

To upload or create a new lookup file, click **New Lookup File**, then click the appropriate option from the drop-down.

Modifying Lookup Files

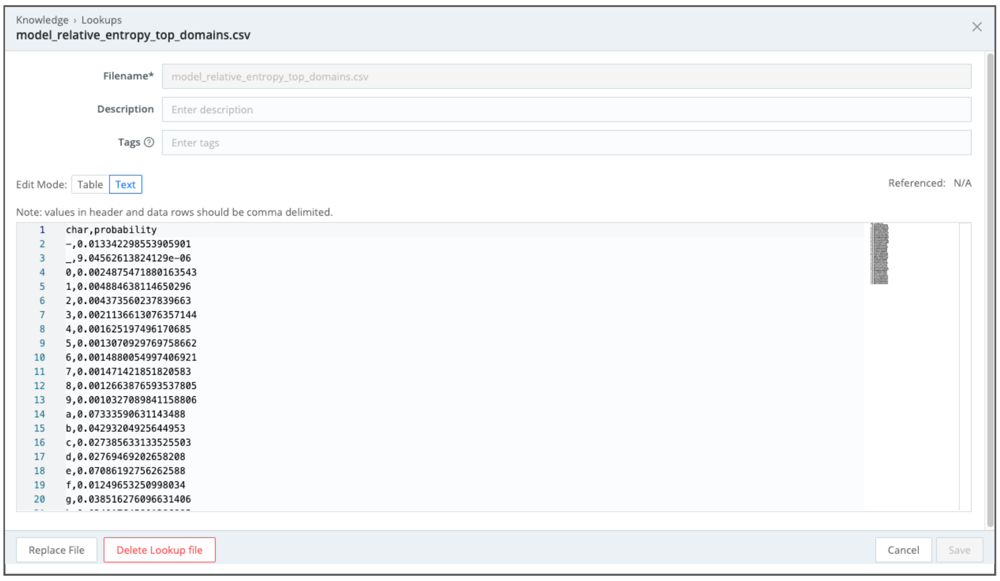
To re-upload, expand, edit, or delete an existing `.csv` or `.gz` lookup file, click its row on the Lookups page. (No editing option is available for `.mmdb` files; you can only re-upload or delete these.)

In the resulting modal, you can edit files in **Table** or **Text** mode. However, **Text** mode is disabled for files larger than 1 MB.



ID	char	probability
1	-	0.013342298553905901
2	_	9.04562613824129e-06
3	0	0.0024875471880163543
4	1	0.004884638114650296
5	2	0.004373560237839663
6	3	0.0021136613076357144
7	4	0.001625197496170685
8	5	0.0013070929769758662
9	6	0.0014880054997406921
10	7	0.001471421851820583

Editing in **Table** mode



Note: values in header and data rows should be comma delimited.

```
char,probability
-,0.013342298553905901
_,9.04562613824129e-06
0,0.0024875471880163543
1,0.004884638114650296
2,0.004373560237839663
3,0.0021136613076357144
4,0.001625197496170685
5,0.0013070929769758662
6,0.0014880054997406921
7,0.001471421851820583
8,0.0012663876593537805
9,0.0010327089841158806
a,0.07333590631143488
b,0.04293204925644953
c,0.027385633133525503
d,0.02769469202658208
e,0.07086192756262588
f,0.01249653250998034
g,0.038516276096631406
```

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 Stream 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 Stream 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:

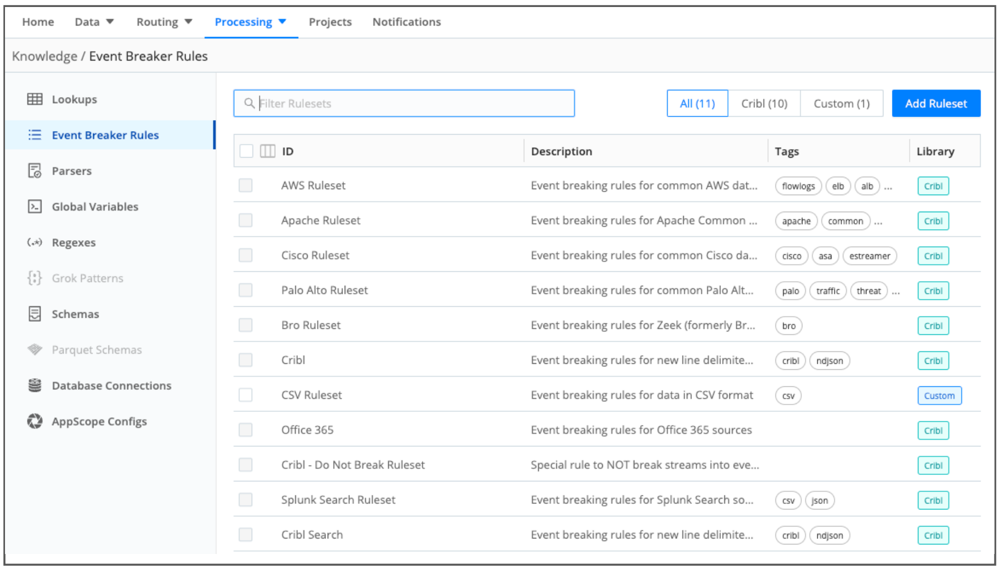
- Lookup Function.
- Ingest-time Lookups use case.
- Managing Large Lookups use case.
- Redis Function for faster lookups using a Redis integration (bypasses the Lookups Library).

18.2. Event Breakers

Event Breakers help break incoming streams of data into discrete events. To see how Event Breakers interact with the rest of Cribl Stream's data flow, see Event Processing Order.

Accessing Event Breakers

You access the Event Breakers management interface from Cribl Stream'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.



ID	Description	Tags	Library
AWS Ruleset	Event breaking rules for common AWS dat...	flowlogs elb alb ...	Cribl
Apache Ruleset	Event breaking rules for Apache Common ...	apache common ...	Cribl
Cisco Ruleset	Event breaking rules for common Cisco da...	cisco asa estreamer	Cribl
Palo Alto Ruleset	Event breaking rules for common Palo Alt...	palo traffic threat ...	Cribl
Bro Ruleset	Event breaking rules for Zeek (formerly Br...	bro	Cribl
Cribl	Event breaking rules for new line delimite...	cribl ndjson	Cribl
CSV Ruleset	Event breaking rules for data in CSV format	csv	Custom
Office 365	Event breaking rules for Office 365 sources		Cribl
Cribl - Do Not Break Ruleset	Special rule to NOT break streams into eve...		Cribl
Splunk Search Ruleset	Event breaking rules for Splunk Search so...	csv json	Cribl
Cribl Search	Event breaking rules for new line delimite...	cribl ndjson	Cribl

Event Breaker Rulesets page

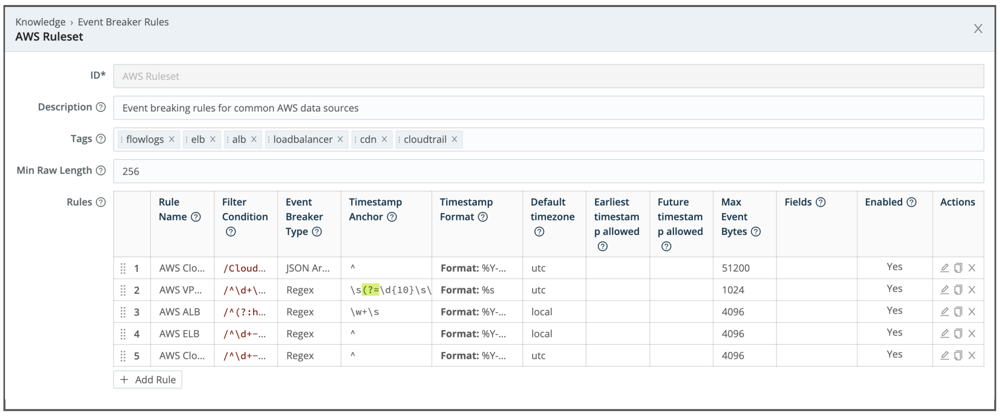
Limitations

Event Breakers are directly accessible only on Sources that require incoming events to be broken into a better-defined format. (Check individual Cribl Stream 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.


Knowledge › Event Breaker Rules
AWS Ruleset

ID*: AWS Ruleset
Description: Event breaking rules for common AWS data sources
Tags: flowlogs, elb, alb, loadbalancer, cdn, cloudtrail
Min Raw Length: 256

Rules:

	Rule Name	Filter Condition	Event Breaker Type	Timestamp Anchor	Timestamp Format	Default timezone	Earliest timestamp allowed	Future timestamp allowed	Max Event Bytes	Fields	Enabled	Actions
1	AWS Clo...	/Cloud...	JSON Ar...	^	**Format:** %Y-...	utc			51200		Yes	
2	AWS VP...	/^\d+\...	Regex	\s(?=\d{10}\s\	**Format:** %s	utc			1024		Yes	
3	AWS ALB	/^(?:h...	Regex	\w+\s	**Format:** %Y-...	local			4096		Yes	
4	AWS ELB	/^\d+-...	Regex	^	**Format:** %Y-...	local			4096		Yes	
5	AWS Clo...	/^\d+-...	Regex	^	**Format:** %Y-...	utc			4096		Yes	

+ Add Rule

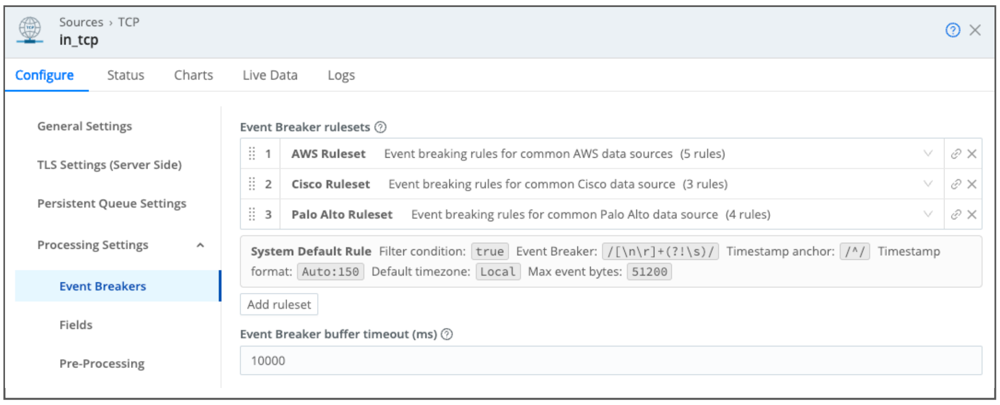
Rules within an example (AWS) ruleset that ships with Cribl Stream

Rules within a ruleset are ordered and evaluated top->down. One or more rulesets can be associated with a Source, and these rulesets are also evaluated top->down. For a stream from a given Source, the first matching rule goes into effect.

Rulesets and Rules - Ordered

```
Ruleset A
  Rule 1
  Rule 2
  ...
  Rule n

...

Ruleset B
  Rule Foo
  Rule Bar
  ...
  Rule FooBar
```

An example of multiple rulesets associated with a Source:



Event Breaker rulesets

1	**AWS Ruleset**	Event breaking rules for common AWS data sources (5 rules)
2	**Cisco Ruleset**	Event breaking rules for common Cisco data source (3 rules)
3	**Palo Alto Ruleset**	Event breaking rules for common Palo Alto data source (4 rules)

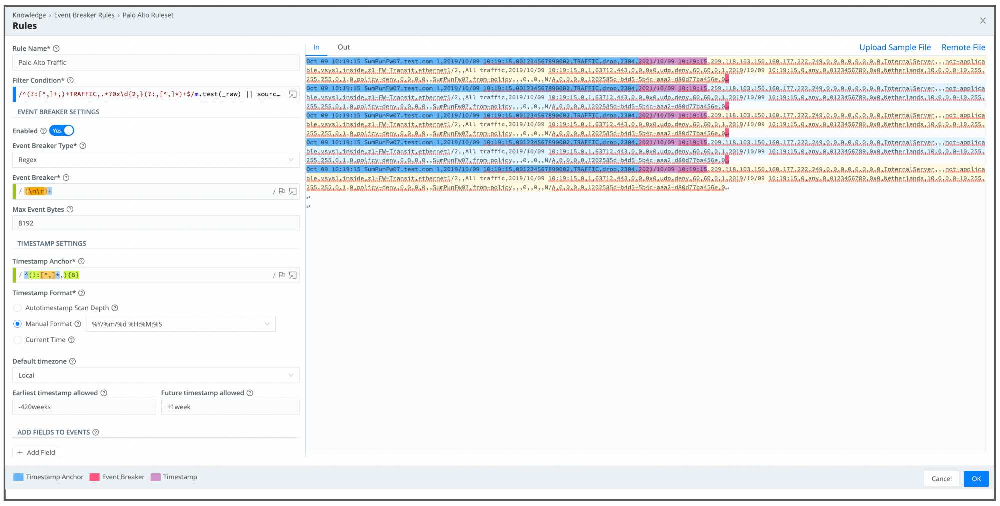
System Default Rule Filter condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp anchor: `/^/` Timestamp format: `Auto:150` Default timezone: `Local` Max event bytes: `51200`

Event Breaker buffer timeout (ms): 10000

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}`.



Knowledge › Event Breaker Rules › Palo Alto Ruleset

Rules

Rule Name*: Palo Alto Traffic

Filter Condition*: `/^(?:[^,]+,)+TRAFFIC,.*?0x\d{2,}(?:,[^,]*)+$/m.test(_raw) || sourc…`

EVENT BREAKER SETTINGS

Enabled: Yes

Event Breaker Type*: Regex

Event Breaker*: `/ [\n\r]+ /`

Max Event Bytes: 8192

TIMESTAMP SETTINGS

Timestamp Anchor*: `/ ^(?:[^,]*,){6} /`

Timestamp Format*: Autotimestamp Scan Depth [✓ Manual Format] Current Time

Manual Format: %Y/%m/%d %H:%M:%S

Default timezone: Local

Earliest timestamp allowed: -420weeks

Future timestamp allowed: +1week

ADD FIELDS TO EVENTS

```
Oct 09 10:19:15 SumPunFw07.test.com 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2021/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applica
ble,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,,SumPunFw07,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
Oct 09 10:19:15 SumPunFw07.test.com 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2021/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applica
ble,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,,SumPunFw07,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
Oct 09 10:19:15 SumPunFw07.test.com 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2021/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applica
ble,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,,SumPunFw07,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
Oct 09 10:19:15 SumPunFw07.test.com 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2021/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applica
ble,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,,SumPunFw07,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
Oct 09 10:19:15 SumPunFw07.test.com 1,2019/10/09 10:19:15,001234567890002,TRAFFIC,drop,2304,2021/10/09 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applica
ble,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,,SumPunFw07,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
```

🟦 Timestamp Anchor 🟥 Event Breaker 🟪 Timestamp

An Event Breaker rule

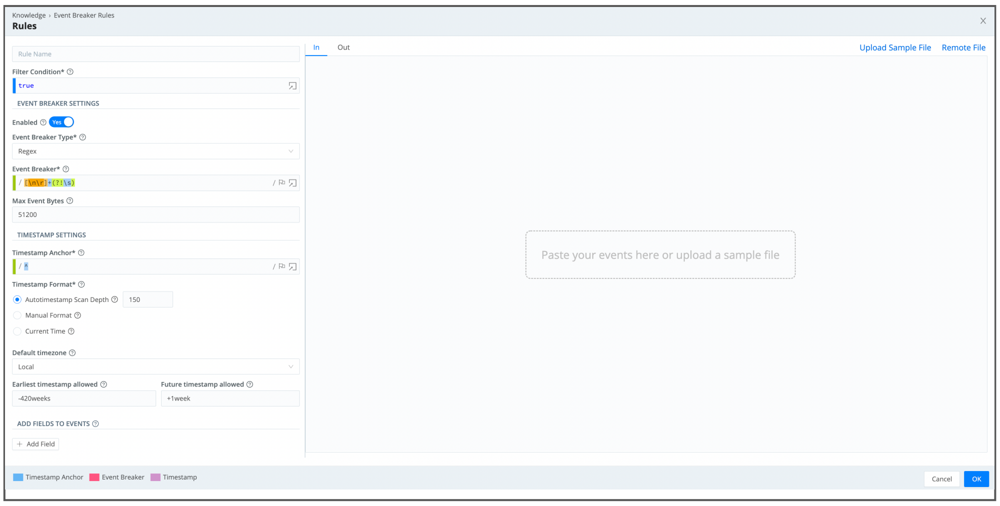
System Default Rule

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.


Knowledge › Event Breaker Rules
Rules

Rule Name

Filter Condition*
`true`

EVENT BREAKER SETTINGS

Enabled: Yes

Event Breaker Type*: Regex

Event Breaker*: `/[\n\r]+(?!\s)/`

Max Event Bytes: 51200

TIMESTAMP SETTINGS

Timestamp Anchor*: `/^/`

Timestamp Format*
[✓ Autotimestamp Scan Depth] 150
Manual Format
Current Time

Default timezone: Local

Earliest timestamp allowed: -420weeks

Future timestamp allowed: +1week

ADD FIELDS TO EVENTS

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

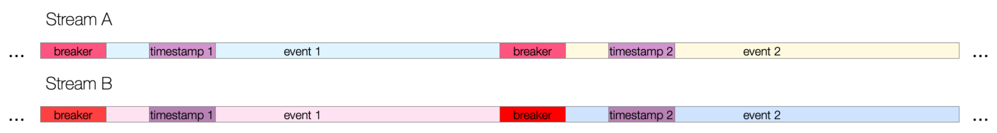
After a breaker pattern has been selected, it will apply on the stream **continuously**. See below for specific information on different Event Breaker Types.

Timestamp Settings

After events are synthesized out of streams, Cribl Stream 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 Stream truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

In Cribl Stream 3.4.2 and above, where an Event Breaker has set an event's `_time` to the current time – rather than extracting the value from the event itself – it will mark this by adding the internal field `__timestampExtracted: false` to the event.

Add Fields to Events

After events have been timestamped, one or more fields can be added here as key-value pairs. In each field's **Value Expression**, you can fully evaluate the field value using JavaScript expressions.

Event Breakers **always** add the `cribl_breaker` field to output events. Its value is the name of the chosen ruleset. (Some examples below omit the `cribl_breaker` field for brevity, but in real life the field is always added.)

Max Event Bytes

If an event's final broken chunk reaches a matched rule's **Max event bytes** length, it will be broken again.

The highest **Max Event Bytes** value that you can set is about 128 MB (`134217728` bytes). Events exceeding that size will be split into separate events, but left unbroken. Cribl Stream will set these events' `__isBroken` internal field to `false`.

Event Breaker Types

Several types of Event Breaker can be applied to incoming data streams:

- Regex

- File Header
- JSON Array
- JSON New Line Delimited
- Timestamp
- CSV

Regex

The Regex breaker uses regular expressions to find breaking points in data streams.

After a breaker regex pattern has been selected, it will apply on the stream **continuously**. Breaking will occur at the beginning of the match, and the matched content will be consumed/thrown away. If necessary, you can use a positive lookahead regex to keep the content – e.g.: `(?=pattern)`

Capturing groups are **not allowed** to be used anywhere in the Event Breaker pattern, as they will further break the stream – which is often undesirable. Breaking will also occur if the final broken event's length reaches the rule's **Max event bytes**.

Example

Break after a newline or carriage return, but only if followed by a timestamp pattern:
Event Breaker: `[\n\r]+(?=\d+-\d+-\d+\s\d+:\d+:\d+)`

Sample Event - Multiline

```
--- input ---
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can pr
   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 pr
   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
  "_time": 1589920332
}

--- output event 2 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then
  "_time": 1589920332
}
```

File Header

You can use the File Header breaker to break files with headers, such as IIS or Bro logs. This type of breaker relies on a header section that lists field names. The header section is typically present at the top of the file, and can be single-line or greater.

After the file has been broken into events, fields will also be extracted, as follows:

- **Header Line**: Regex matching a file header line. For example, `^#`.
- **Field Delimiter**: Field delimiter regex. For example, `\s+`.
- **Field Regex**: Regex with one capturing group, capturing all the fields to be broken by field delimiter. For example, `^#[Ff]ields[:]?\s+(.*)`
- **Null Values**: Representation of a null value. Null fields are not added to events.
- **Clean Fields**: Whether to clean up field names by replacing non `[a-zA-Z0-9]` characters with `_`.

Example

Using the values above, let’s see how this sample file breaks up:

Sample Event - File Header

```
--- input ---
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
#types  time    string  addr    port    addr    port    enum
1331904608.080000       -     192.168.204.59  137     192.168.204.255 137     udp
1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53      udp


--- output event 1 ---
{
  "_raw": "1331904608.080000       -     192.168.204.59  137     192.168.204.255 13
  "ts": "1331904608.080000",
  "id_orig_h": "192.168.204.59",
  "id_orig_p": "137",
  "id_resp_h": "192.168.204.255",
  "id_resp_p": "137",
  "proto": "udp",
  "_time": 1331904608.08
}

--- output event 2 ---
{
  "_raw": "1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53
  "ts": "1331904609.190000",
  "id_orig_h": "192.168.202.83",
  "id_orig_p": "48516",
  "id_resp_h": "192.168.207.4",
  "id_resp_p": "53",
  "proto": "udp",
  "_time": 1331904609.19
}
```

JSON Array

You can use the JSON Array to extract events from an array in a JSON document (e.g., an Amazon CloudTrail file).

- **Array Field**: Optional path to array in a JSON event with records to extract. For example, `Records`.
- **Timestamp Field**: Optional path to timestamp field in extracted events. For example, `eventTime` or `level1.level2.eventTime`.

- **JSON Extract Fields**: Enable this toggle to auto-extract fields from JSON events. If disabled, only `_raw` and `time` will be defined on extracted events.
- **Timestamp Format**: If **JSON Extract Fields** is set to `No`, you **must** set this to **Autotimestamp** or **Current Time**. If **JSON Extract Fields** is set to `Yes`, you can select any option here.

Example

Using the values above, let's see how this sample file breaks up:

Sample Event - JSON Document (Array)

```
--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:55Z","eventSource"
{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:56Z","eventSource":"ec2.amazo

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:55Z\",\"eve
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:56Z\",\"eve
  "_time": 1586309756,
  "cribl_breaker": "j-array"
}
```

JSON New Line Delimited

You can use the JSON New Line Delimited breaker to break and extract fields in newline-delimited JSON streams.

Example

Using default values, let's see how this sample stream breaks up:

Sample Event - Newline Delimited JSON Breaker

```
--- input ---
{"time":"2020-05-25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"inf
{"time":"2020-05-25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"inf


--- output event 1 ---
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"clus
  "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\":\"clus
  "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 Stream finds a timestamp. This type enables breaking on lines whose timestamp pattern is not known ahead of time.

Example

Using default values, let’s see how this sample stream breaks up:

Sample Event - Timestamp Based Breaker

```
--- input ---
{"level":"debug","ts":"2021-02-02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"H
{"level":"debug","ts":"2021-02-02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"H


--- output event 1 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:46.365Z\",\"caller\":\"sd
  "_time": 1612262326.365
}

--- output event 2 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-02T10:38:56.365Z\",\"caller\":\"sd
  "_time": 1612262336.365
}
```

CSV

The CSV breaker extracts fields in CSV streams that include a header line. Selecting this type exposes these extra fields:

- **Delimiter**: Delimiter character to use to split values. Defaults to: `,`
- **Quote Char**: Character used to quote literal values. Defaults to: `"`
- **Escape Char**: Character used to escape the quote character in field values. Defaults to: `"`
  Example: Using default values, let’s see how this sample stream breaks up:

Example

Using default values, let’s see how this sample stream breaks up:

Sample Event - CSV Breaker

```
--- input ---
time,host,source,model,serial,bytes_in,bytes_out,cpu
1611768713,"myHost1","anet","cisco","ASN4204269",11430,43322,0.78
1611768714,"myHost2","anet","cisco","ASN420423",345062,143433,0.28


--- output event 1 ---
{
  "_raw": "\"1611768713\",\"myHost1\",\"anet\",\"cisco\",\"ASN4204269\",\"11430\",\
  "time": "1611768713",
  "host": "myHost1",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN4204269",
  "bytes_in": "11430",
  "bytes_out": "43322",
  "cpu": "0.78",
  "_time": 1611768713
}

--- output event 2 ---
{
  "_raw": "\"1611768714\",\"myHost2\",\"anet\",\"cisco\",\"ASN420423\",\"345062\",\
  "time": "1611768714",
  "host": "myHost2",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN420423",
  "bytes_in": "345062",
  "bytes_out": "143433",
  "cpu": "0.28",
  "_time": 1611768714
}
```

With **Type: CSV** selected, an Event Breaker will properly add quotes around all values, regardless of their initial state.

Cribl Versus Custom Rulesets

Event Breaker rulesets shipped by Cribl will be listed under the **Cribl** tag, while user-built rulesets will be listed under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search.

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 Stream 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 Stream 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.

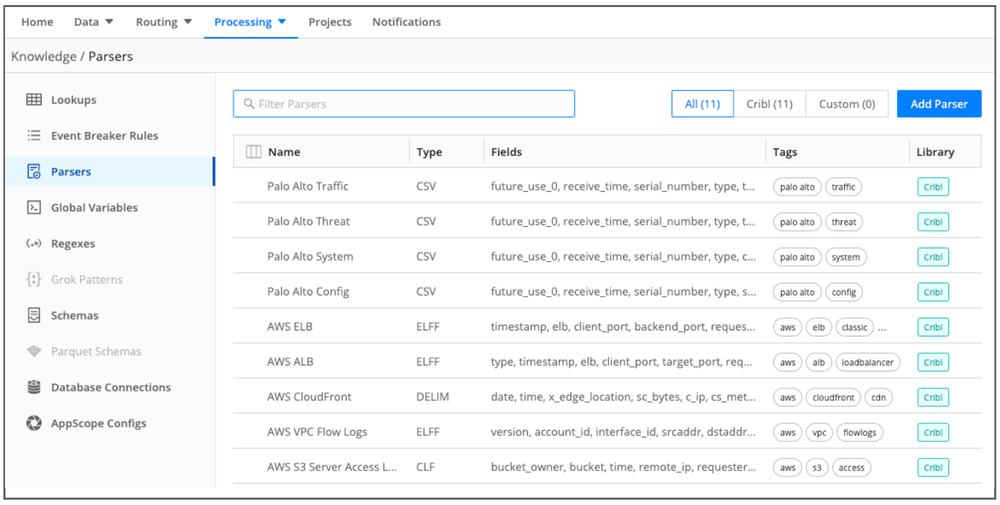
When determining which Event Breaker to use, Cribl Stream creates a substring from the incoming `_raw` field, at the length set by rules' **Max Event Bytes** parameters. If Cribl Stream does not get a match, it checks whether the substring of `_raw` has at least the threshold number of characters set by **Min Raw Length**. If so, Cribl Stream 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).

18.3. Parsers Library

What Are Parsers

Parsers are definitions and configurations for the Parser Function. You can find the library from Cribl Stream'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.



Name	Type	Fields	Tags	Library
Palo Alto Traffic	CSV	future_use_0, receive_time, serial_number, type, t...	palo alto, traffic	Cribl
Palo Alto Threat	CSV	future_use_0, receive_time, serial_number, type, t...	palo alto, threat	Cribl
Palo Alto System	CSV	future_use_0, receive_time, serial_number, type, c...	palo alto, system	Cribl
Palo Alto Config	CSV	future_use_0, receive_time, serial_number, type, s...	palo alto, config	Cribl
AWS ELB	ELFF	timestamp, elb, client_port, backend_port, reques...	aws, elb, classic ...	Cribl
AWS ALB	ELFF	type, timestamp, elb, client_port, target_port, req...	aws, alb, loadbalancer	Cribl
AWS CloudFront	DELIM	date, time, x_edge_location, sc_bytes, c_ip, cs_met...	aws, cloudfront, cdn	Cribl
AWS VPC Flow Logs	ELFF	version, account_id, interface_id, srcaddr, dstaddr...	aws, vpc, flowlogs	Cribl
AWS S3 Server Access L...	CLF	bucket_owner, bucket, time, remote_ip, requester...	aws, s3, access	Cribl

Parsers Library

Parsers can be used to **extract** or **reserialize** events. See Parser Function page for examples.

Supported Parser Types:

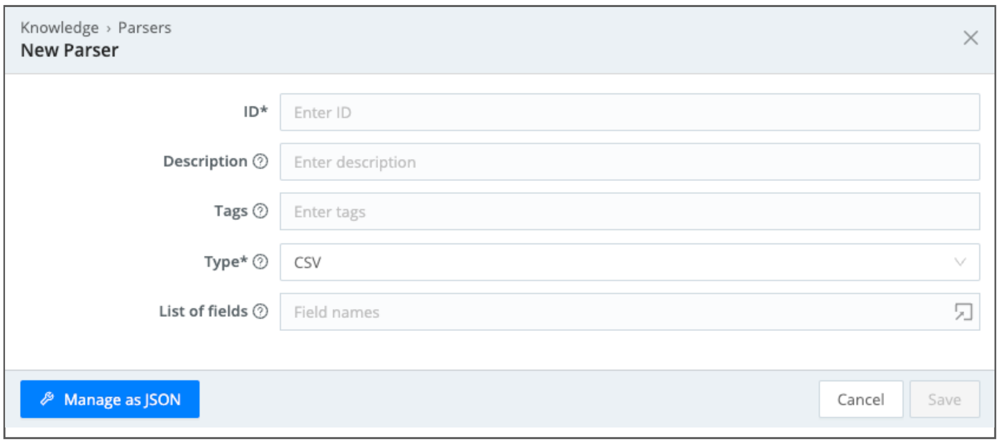
- CSV – Parse and reserialize comma-separated values.
- ELFF – Parse and reserialize events in Extended Log File Format.
- CLF – Parse and reserialize events in Common Log Format.

Creating a Parser

To create a parser, follow these steps:

1. Go to **Knowledge** > **Parsers** and click **Add Parser**.
2. Enter a unique **ID**.
3. Optionally, enter a **Description**.
4. Select a **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

18.4. 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 Stream'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 Stream 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 Stream 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 Stream by default.
- Global Variable Value: `Date.now()/1000`
- **Sample Eval Function:** `nowEpoch = C.vars.epoch()`

Scenario 3:

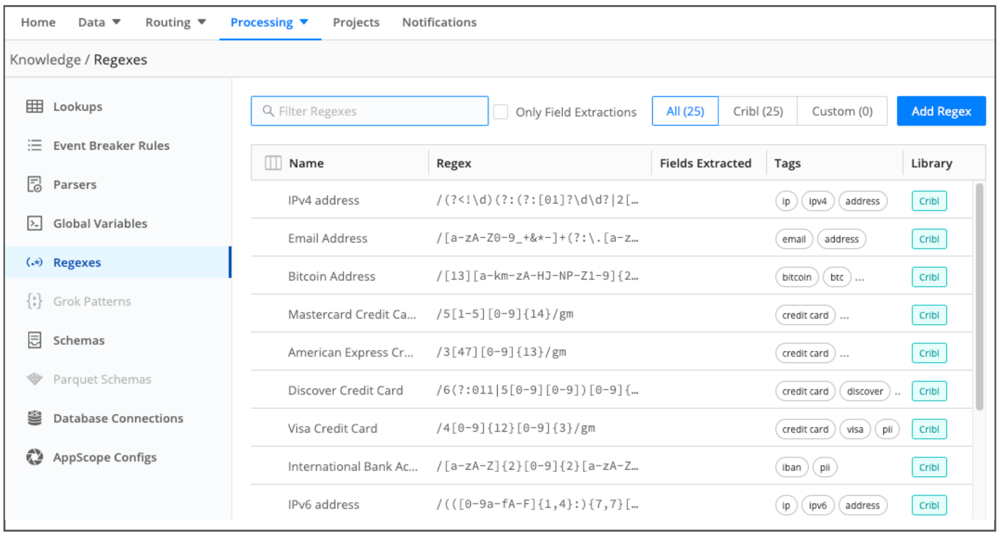
Create a new field called `storage`, by converting the value of event field `size` to human-readable format.

- Global Variable Name: `convertBytes` <– ships with Cribl Stream 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.

18.5. Regex Library

What Is the Regex Library

Cribl Stream 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 Stream's top nav under **Processing** > **Knowledge** > **Regexes** .

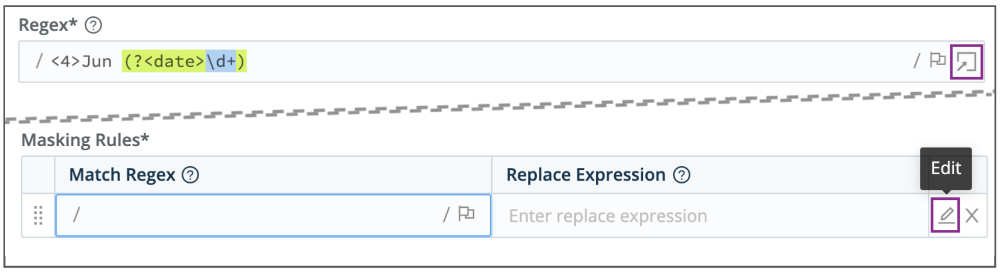


Name	Regex	Fields Extracted	Tags	Library
IPv4 address	/(?<!\d)(?:(?:[01]?\d\d?|2[…		ip ipv4 address	Cribl
Email Address	/[a-zA-Z0-9_+&*-]+(?:\.[a-z…		email address	Cribl
Bitcoin Address	/[13][a-km-zA-HJ-NP-Z1-9]{2…		bitcoin btc ...	Cribl
Mastercard Credit Ca...	/5[1-5][0-9]{14}/gm		credit card ...	Cribl
American Express Cr...	/3[47][0-9]{13}/gm		credit card ...	Cribl
Discover Credit Card	/6(?:011|5[0-9][0-9])[0-9]{…		credit card discover ..	Cribl
Visa Credit Card	/4[0-9]{12}[0-9]{3}/gm		credit card visa pii	Cribl
International Bank Ac...	/[a-zA-Z]{2}[0-9]{2}[a-zA-Z…		iban pii	Cribl
IPv6 address	/(([0-9a-fA-F]{1,4}:){7,7}[…		ip ipv6 address	Cribl

Regular Expression Library

Using Library Patterns

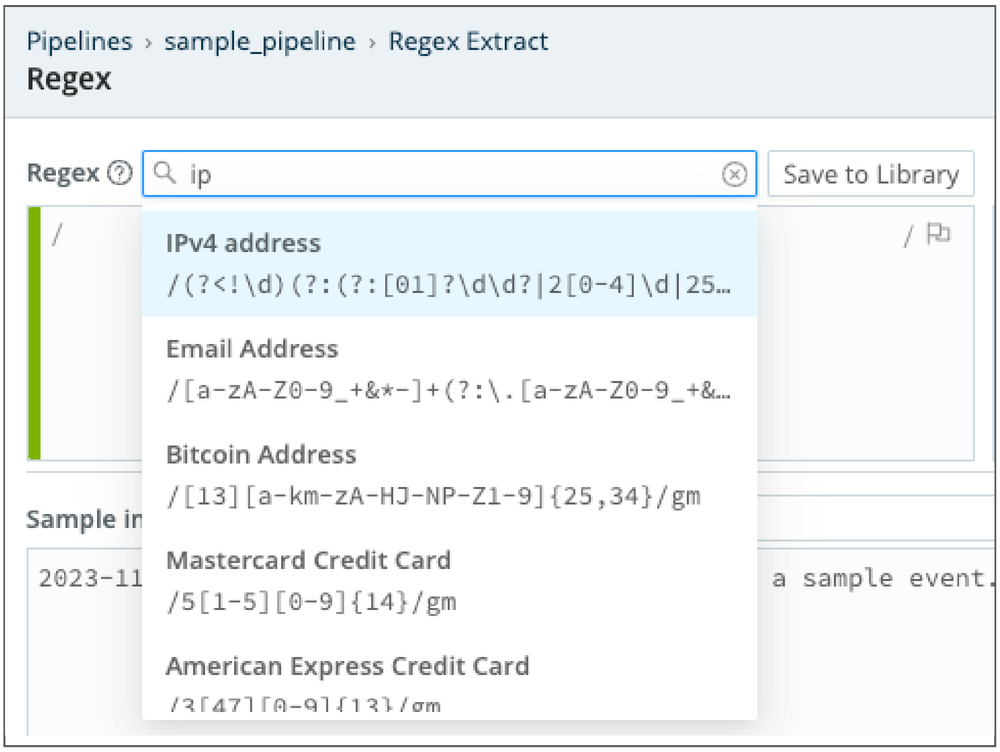
As of this version, the Library contains 25 patterns shipped by Cribl Stream. To insert a pattern into a Function's regex field, first click the pop-out or Edit icon beside that field.



```
/ <4>Jun (?<date>\d+) /
```

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.


Pipelines › sample_pipeline › Regex Extract
Regex

Regex ⓘ ip   Save to Library

IPv4 address
```
/(?<!\d)(?:(?:[01]?\d\d?|2[0-4]\d|25…
```
Email Address
```
/[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&…
```
Bitcoin Address
```
/[13][a-km-zA-HJ-NP-Z1-9]{25,34}/gm
```
Mastercard Credit Card
```
/5[1-5][0-9]{14}/gm
```
American Express Credit Card

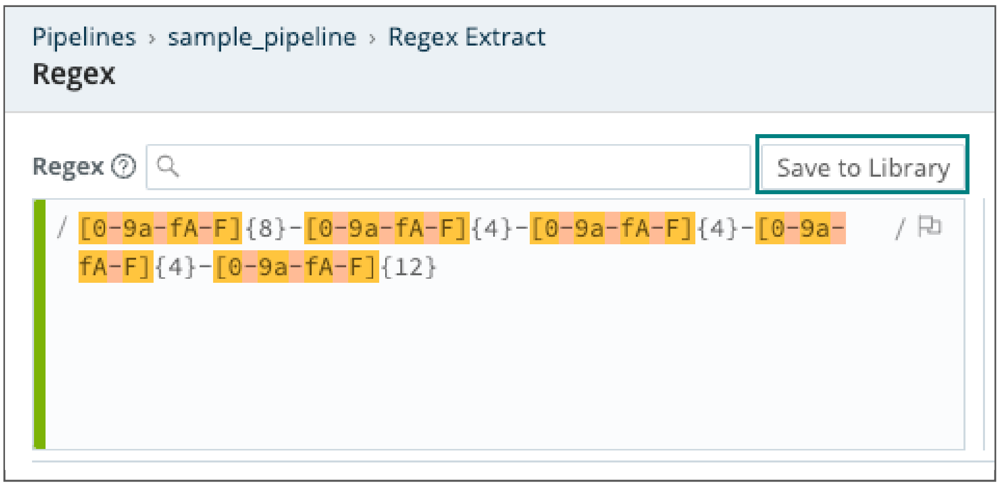
Sample in

```
2023-11                      a sample event.
```

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.

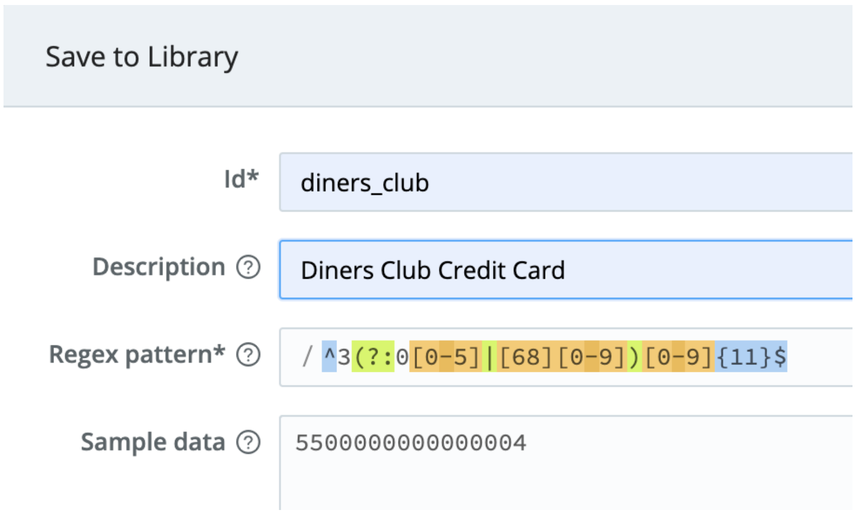

Pipelines › sample_pipeline › Regex Extract
Regex

Regex ⓘ   Save to Library

```
/ [0-9a-fA-F]{8}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{4}-[0-9a-fA-F]{12}   /
```

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**.


Save to Library

Id*	diners_club
Description	Diners Club Credit Card
Regex pattern*	/ ^3(?:0[0-5]|[68][0-9])[0-9]{11}$
Sample data	5500000000000004

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

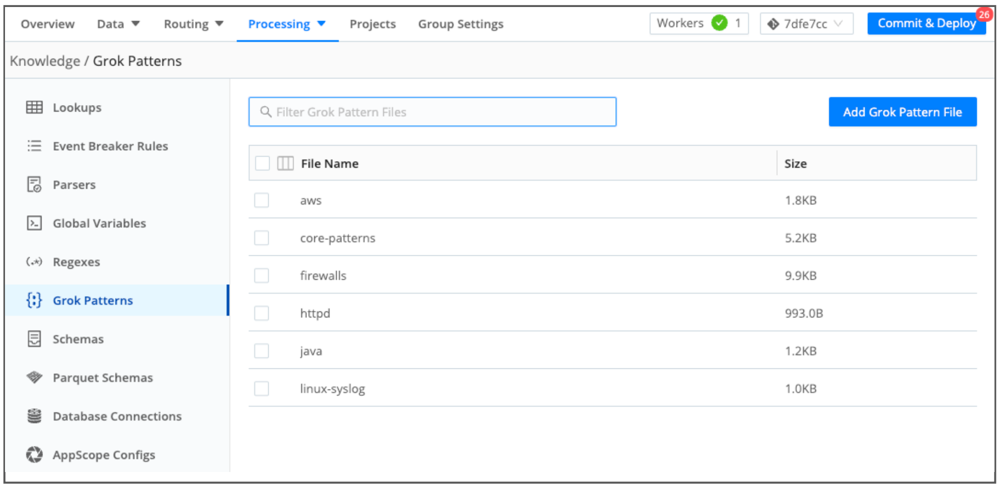
Within the Library, patterns shipped by Cribl will be listed under the **Cribl** tab, while those built by users will be found under **Custom**. Over time, Cribl Stream will ship more patterns, and this distinction allows for both sets to grow independently.

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.

18.6. Grok Patterns Library

What Is the Grok Patterns Library

Cribl Stream ships with a Grok Patterns Library that contains a set of pre-built common patterns, organized as files.



File Name	Size
aws	1.8KB
core-patterns	5.2KB
firewalls	9.9KB
httpd	993.0B
java	1.2KB
linux-syslog	1.0KB

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 **Add Grok Pattern File**. In the resulting modal, assign a unique **Filename**, populate the file with patterns, then click **Save**.

Pattern files reside in: `$CRIBL_HOME/(default|local)/cribl/grok-patterns/`

Using Grok Patterns

In the current Cribl Stream version, you apply Grok patterns by inserting a Grok Function into a Pipeline, then manually typing or pasting patterns into the **Pattern** field(s).

18.7. Schemas Library

Cribl Stream 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 Stream 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 Stream supports schemas matching that standard's Drafts 0 through 7.

You can access the schema library from Cribl Stream'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 Stream 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:

Events

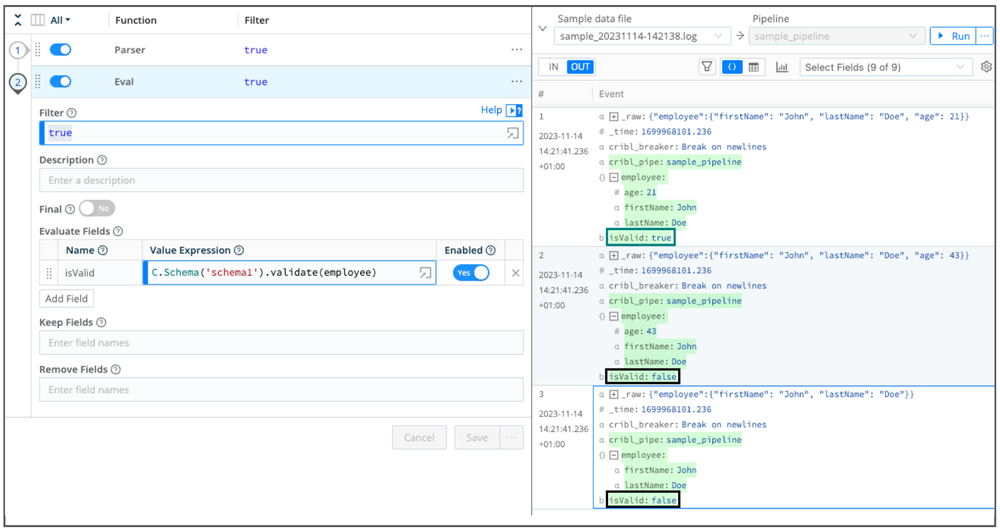
```
{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}
```

To validate whether the `employee` field is valid per `schema1`, we can use the following:
`C.Schema('schema1').validate(employee)`

Results:

- First event is **valid**.
- Second event is **not valid** because `age` is greater than the maximum of `42`.

- Third event is **not valid** because `age` is missing.



Function	Filter
Parser	true
Eval	true

Filter: `true`

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression	Enabled
isValid	`C.Schema('schema1').validate(employee)`	Yes

Keep Fields: Enter field names

Remove Fields: Enter field names

Sample data file: sample_20231114-142138.log → Pipeline: sample_pipeline

```
#  Event
1  _raw: {"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
   _time: 1699968101.236
   cribl_breaker: Break on newlines
   cribl_pipe: sample_pipeline
   employee:
     age: 21
     firstName: John
     lastName: Doe
   isValid: true
2023-11-14 14:21:41.236 +01:00

2  _raw: {"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
   _time: 1699968101.236
   cribl_breaker: Break on newlines
   cribl_pipe: sample_pipeline
   employee:
     age: 43
     firstName: John
     lastName: Doe
   isValid: false
2023-11-14 14:21:41.236 +01:00

3  _raw: {"employee":{"firstName": "John", "lastName": "Doe"}}
   _time: 1699968101.236
   cribl_breaker: Break on newlines
   cribl_pipe: sample_pipeline
   employee:
     firstName: John
     lastName: Doe
   isValid: false
2023-11-14 14:21:41.236 +01:00
```

Schema validation results for the above events

18.8. Parquet Schemas

Cribl Stream supports two kinds of schemas:

- Parquet schemas for writing data from a Cribl Stream 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.

When logging is set to `debug`, you will see what schema is associated with the Parquet files that Cribl Stream writes out. On Linux, you can use the Cribl Stream CLI's `parquet` command to view a Parquet file, its metadata, or its schema.

If you turn on a Destination's **Parquet Settings** > **Automatic schema**, Cribl Stream will automatically generate a Parquet schema based on the events of each Parquet file it writes. The automatically-generated schema preserves the exact structure of the ingested events, with no data lost. In Cribl Stream v.4.4.2, every field in the generated schema will have a data type of `JSON`, `STRING`, `INT_64`, `UNSIGNED_INT_64`, or `DOUBLE`.

(The one exception is the Amazon Security Lake Destination, where automatic schema generation is not relevant because Amazon Security Lake exclusively uses the OCSF Parquet schemas already available in the drop-down.)

Pros and Cons of Automatically Generated Parquet schemas

When you decide whether or not to generate Parquet schemas automatically, consider these pros and cons.

Advantages:

- Configuration is minimal.
- There's no need to know how events you want to ingest are structured.

- There's no need to change Parquet schema when event structure changes.

Disadvantages:

- Writing Parquet files takes longer than with predefined schemas. Note that schema complexity increases linearly with the number of events used to generate the schema.
- Automatically-generated Parquet schemas cannot be as fine-tuned as predefined ones, to capture certain nuances. For example, encoding will always be `PLAIN`, compression will always be `SNAPPY`.
- Some aspects of event structure (such as maps and sets) cannot be captured in the schema.
- Some aspects of time-related fields cannot be captured in the schema.
- Because the system produces a new Parquet schema for each file written, there may be differences between successive schemas. This can limit your ability to concatenate a collection of Parquet files.

Configuring Parquet Schemas

If you are not using automatic schema generation, or if there's no pre-existing schema that suits your data:

- Before configuring a Destination for Parquet output, you should add an existing Parquet schema, or create a new Parquet schema that suits the data you're working with.
- You do not need to start from scratch: Cribl provides sample Parquet schemas for you to clone and then customize as needed.

From the top nav, select **Processing** > **Knowledge**, then select the **Parquet Schemas** left tab.

If you're adding a Parquet schema:

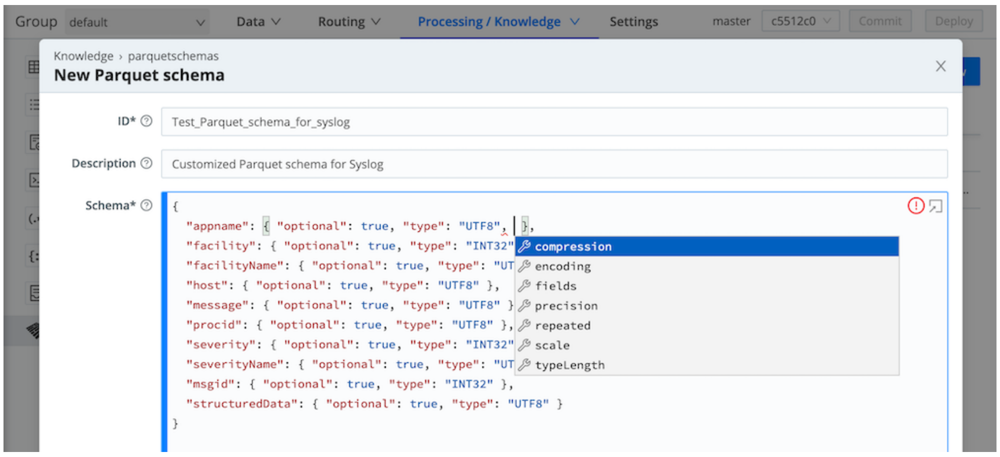
- Click **Add Parquet schema** to open the **New Parquet schema** modal.

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**.



New Parquet schema

ID*: Test_Parquet_schema_for_syslog

Description: Customized Parquet schema for Syslog

Schema*:

```
{
  "appname": { "optional": true, "type": "UTF8", | },
  "facility": { "optional": true, "type": "INT32"
  "facilityName": { "optional": true, "type": "UT
  "host": { "optional": true, "type": "UTF8" },
  "message": { "optional": true, "type": "UTF8" }
  "procid": { "optional": true, "type": "UTF8" },
  "severity": { "optional": true, "type": "INT32"
  "severityName": { "optional": true, "type": "UT
  "msgid": { "optional": true, "type": "INT32" },
  "structuredData": { "optional": true, "type": "UTF8" }
}
```

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 Stream

Different Parquet readers and writers behave differently. Keep the following guidelines in mind when working with Parquet in Cribl Stream.

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 Stream currently does not fully support autocompletion on deeply nested schemas.)

File Extensions

For now, Cribl Stream can read Parquet files only if they have the extension `.parquet`, `.parq`, or `.pqt`.

Field Content

When Cribl Stream writes to a Parquet file:

- If the data contains a field that **is not** present in the schema – i.e, an extra field – Cribl Stream 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 Stream treats this as a data mismatch. Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream does not support reading or writing Parquet files which employ the deprecated `INT96` data type.

18.9. Database Connections

You configure Database Connections to define DBMS resources from which the Database Collector can retrieve events.

Accessing Database Connections

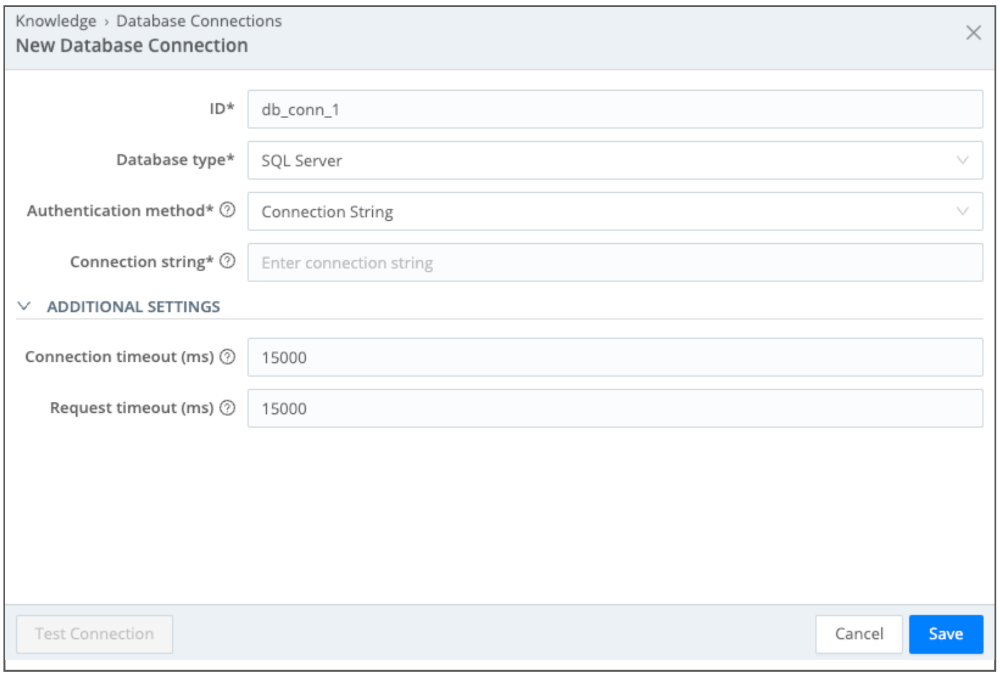
You access Database Connections from Cribl Stream's top nav under **Processing** > **Knowledge** > **Database Connections**. Select **New Database Connection** to open a modal that provides the following controls.

Initial Settings

ID: Enter a unique ID for this connection. (A descriptive ID will help all users recognize the connection by purpose.)

Database type: Select from the drop-down options. The currently supported options are `MySQL`, `SQL Server`, and `Postgres`.

The **Database type** that you select exposes slightly different configuration fields and compatible values, listed below.



Knowledge › Database Connections
New Database Connection

ID*	db_conn_1
Database type*	SQL Server
Authentication method*	Connection String
Connection string*	Enter connection string

ADDITIONAL SETTINGS

Connection timeout (ms)	15000
Request timeout (ms)	15000

New Database Connection modal

Authentication Settings

Set the **Authentication method** to one of the following options.

Connection String: Exposes a field where you directly enter your database connection string. The string's format varies according to your selected **Database type**.

- MySQL format: `mysql[s]://[[user][:password@]][host][:port][/db][?option=value]`
- SQL Server format: `Server=localhost; Database=test;User Id=SA; Password=Testing123! [; Option=Value]`
- Postgres format: `postgres[ql]://[userspec@][hostspec][/dbname][?paramspec]`

Secret: Exposes a **Connection string secret** drop-down, where you select an existing stored text secret that provides your connection string in the appropriate format listed above. A **Create** button is available to define a new secret.

Config: When **Database type** is set to `SQL Server`, exposes a pane where you can enter a JSON connection configuration object. The object uses the Tedious driver and the node-mssql client.

Additional Settings

Connection timeout (ms): How long, in milliseconds, Cribl Stream should wait before assuming that a connection has failed. This defaults to `10000` (10 sec.) for MySQL databases, and to `15000` (15 sec.) for SQL Server databases. For both, the minimum allowed value is `1000` (1 sec.), and the maximum is `60000` (1 minute).

Request timeout (ms): This setting appears only where **Database type** is set to `SQL Server`. Defines how long Cribl Stream should wait before assuming that a request has failed. If a query requires a longer timeout than the default `15000` (15 seconds), be sure that you know why – long-running queries can affect other parts of the system. Minimum is `1000` (1 second); maximum is `600000` (10 minutes).

Testing the Connection

To test database configuration, click **Test Connection** at the modal's lower left. This will perform the test from the Leader Node. To test a connection from a Worker Node, teleport to the Worker first.

Examples

To adapt the following examples to your deployment, substitute your own credentials, domain names, etc, for those in the examples.

Connecting to a MySQL Database

- Database type: `MySQL`
- Connection String: `mysqls://userName:pa$$w0rd@server:3306/dbName`

Using Active Directory to Connect to SQL Server on Azure

- Database type: `SQL Server`
- Connection String: `Server=tcp:server.database.windows.net,1433;User ID=userName@domain.onmicrosoft.com;Password=pa$$w0rd;Encrypt=True;TrustServerCerti Directory Password";Database="dbName"`

Using NTLM to Connect to SQL Server

Windows Challenge/Response, also known as Microsoft New Technology LAN Manager (NTLM), is an authentication protocol for systems running the Windows operating system.

- Database type: `SQL Server`
- Configuration Object:

```
{
  "authentication": {
    "type": "ntlm",
    "options": {
      "userName": "userName",
      "password": "pa$$w0rd",
      "domain": "domain"
    }
  },
  "options": {
    "connectTimeout": 15000,
    "trustServerCertificate": true
  },
  "connectionTimeout": 15000,
  "port": 1433,
  "server": "server",
  "database": "database"
}
```

18.10. AppScope Configs

You can obtain high-fidelity application data from any Linux command or application using AppScope, an open-source tool from Cribl. This includes data that's difficult or impossible to obtain any other way. (Note that AppScope is no longer being actively developed by Cribl.)

AppScope is included in your Cribl Stream installation. One or two AppScope Sources are enabled right out-of-the-box; they are configured to send data to Cribl Edge or Cribl Stream on the same host.

AppScope ships with a set of config files that offer a variety of options for what events and metrics you want AppScope to capture, and whether to enable payload capture. These are: `sample_config`, `default_metrics_events`, `verbose_metrics_events`, and `verbose_metrics_events_payloads`.

To understand precisely what one of these files does, select it to open its **AppScope Configs** modal and click **Manage as JSON**. For explanations and definitions, see the Config File and Schema Reference sections in the AppScope docs.

You can customize the config files, or edit them via **Manage as JSON**. When editing, you can revert to the original state by clicking **Restore Default**. The safer option is to click **Clone AppScope config**, then edit and save your new version with a new filename. The sample configs, and any variants you have created and saved, constitute a reusable set of AppScope configurations in the AppScope Config Library.

When you have assigned an AppScope config to a Rule in an AppScope Source, and you change it in the Knowledge UI, the changes will not take effect until you do the following:

- In the AppScope Source **AppScope Rules** settings, re-assign the config to the Rule. (That is, unselect the config file, then select it again.)
- Save and commit.

Creating AppScope Configs

You access the **AppScope Configs** UI from Cribl Stream'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: `^240100000000000000000000d407`

Convert to hex: `Yes`

Convert length: `14`

HTTP

By default, AppScope uses an internally-defined protocol detector for HTTP similar to this example.

Name: `HTTP`

Regex: `"HTTP\\/1\\.[0-2]|PRI \\* HTTP\\/2\\.0\r\n\r\nSM\r\n\r\n"`

StatsD

By default, AppScope uses an internally-defined protocol detector for StatsD similar to this example.

Name: `STATSD`

Regex: `"^([^:]+):([\\d.]+)\\|(c|g|ms|s|h)"`

TLS

By default, AppScope uses an internally-defined protocol detector for TLS/SSL similar to this example.

Name: `TLS`

Regex: `"^(?:(?:16030[0-3].{4})|(?:8[0-9a-fA-F]{3}01))"`

Convert to hex: `Yes`

Convert length: `5`

19. Reference

19.1. API Docs

To complement our API Reference, below are some examples of using the Cribl Stream API to address common scenarios. If you want to automate an action in Cribl Stream for which you can't find documentation, ask us in Cribl Community's `#docs` channel.

Breaking Change – v.4.0+

In Cribl Stream 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 Worker Group, 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 Stream instance is deployed, as outlined in this section. You can make the listed requests at the command line, programmatically, or using the UI.

On-Prem

To obtain a Bearer token, send a local authentication request and payload to the API as shown below. Adapt the example to include your Leader hostname (or IP address), username, and password.

If you're using SSO/OpenID Connect Authentication, you must toggle **Allow local auth** on, because you'll need to be a Local user when you authenticate via the API.

Request:

```
POST https://<cribl-stream-leader>:9000/api/v1/auth/login
```

Payload:

```
{
    "username": "<username>",
    "password": "<password>"
}
```

Example request using `curl`:

```
curl -X POST https://<cribl-stream-leader>:9000/api/v1/auth/login -H "Content-Type
```

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

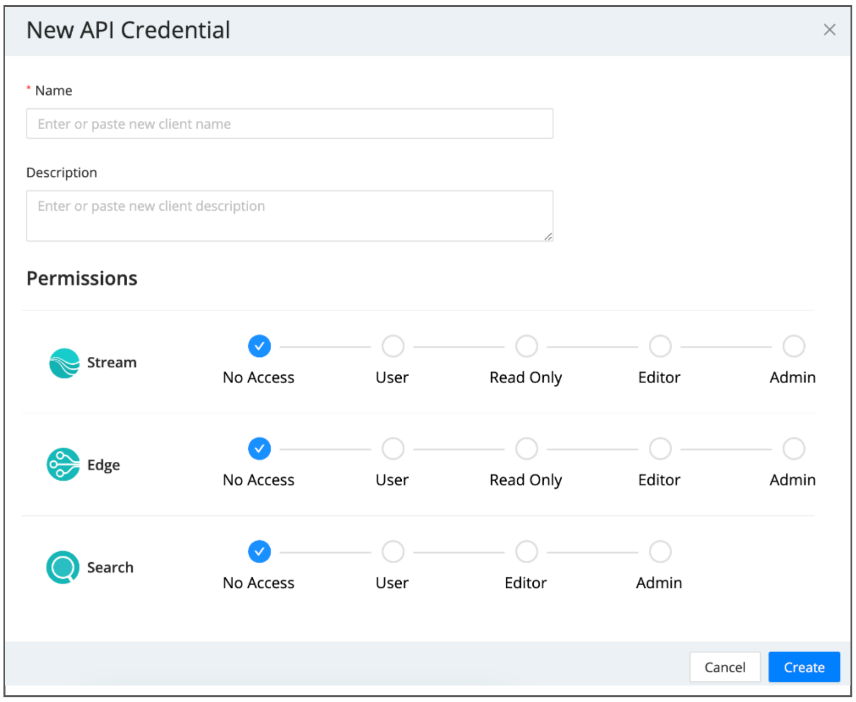
As of Cribl Stream 4.1 (March 2023), Owners and Admins can generate API access tokens directly in their Cribl.Cloud Organization's portal. This is a two-tier process:

- 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 **Permissions** that the Credential's tokens will grant.
   The **Permissions** selector is available with an Enterprise plan. Without an Enterprise plan, all tokens will be granted the Admin Role. See Cribl.Cloud Roles for details on the available permissions.
4. Click **Create** to save the Credential.


New API Credential

* Name

Enter or paste new client name

Description

Enter or paste new client description

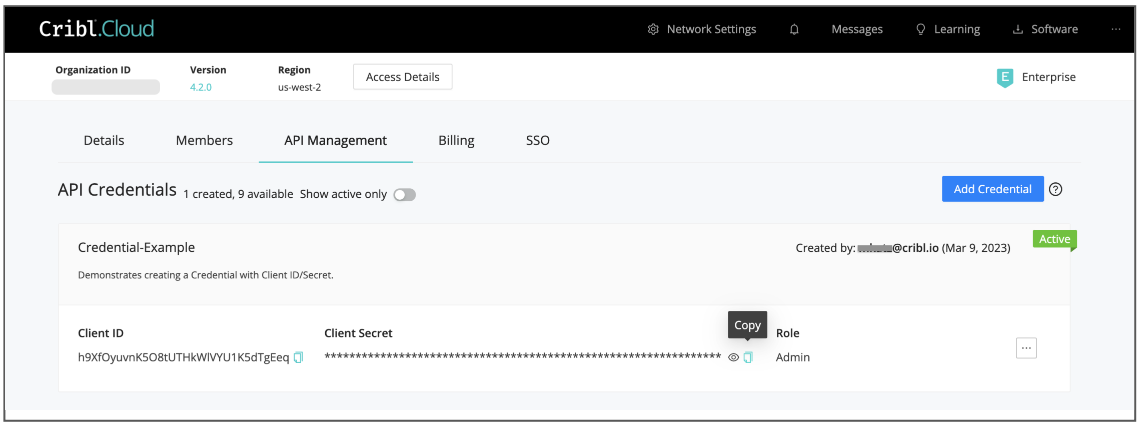
Permissions

Stream	[✓ No Access]	User	Read Only	Editor	Admin
Edge	[✓ No Access]	User	Read Only	Editor	Admin
Search	[✓ No Access]	User	Editor	Admin	

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.



Organization ID: [redacted]
Version: 4.2.0
Region: us-west-2

API Credentials 1 created, 9 available Show active only

Credential-Example

Demonstrates creating a Credential with Client ID/Secret.

Created by: [redacted]@cribl.io (Mar 9, 2023)

Client ID	Client Secret	Role
h9XfOyuvnK5O8tUTHkWlVYU1K5dTgEeq	**	Admin

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_se
```

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:

Single-instance Distributed

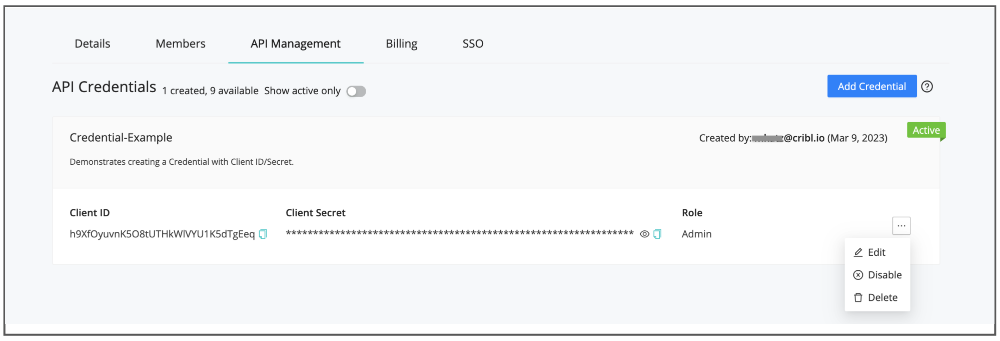
```
curl --location 'https://main-<organizationID>.cribl.cloud/api/v1/m/{workerGrou
--header 'Authorization: Bearer <token-generated-above>'
```

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.



API Credentials 1 created, 9 available Show active only

Credential-Example

Created by: [redacted]@cribl.io (Mar 9, 2023)

Demonstrates creating a Credential with Client ID/Secret.

Client ID	Client Secret	Role
h9XfOyuvnK5O8tUTHkWlVYU1K5dTgEeq	**	Admin

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:

Single-instance   Distributed

```
curl -X GET "<url>/api/v1/m/{workerGroup}/system/inputs/<output id>" -H "accept
```

Example response:

```
{
  "systemFields": [
    "cribl_pipe"
  ],
  "signatureVersion": "v4",
  "objectACL": "private",
  "partitionExpr": "`${host}/${sourcetype}`",
  "format": "json",
  "baseFileName": "CriblOut",
  "compress": "none",
  "maxFileSizeMB": 32,
  "maxFileOpenTimeSec": 300,
  "maxFileIdleTimeSec": 30,
  "maxOpenFiles": 100,
  "onBackpressure": "block",
  "id": "minio",
  "type": "minio",
  "endpoint": "http://minio:9090",
  "bucket": "test",
  "destPath": "keyprefix",
  "stagePath": "tmp",
  "awsApiKey": "minioadmin",
  "awsSecretKey": "minioadmin"
}
```

2. Edit the definition so that when you send it back in a `PATCH` request, it updates the desired Destination's S3 keys.
    - Edit the value of the `id` field to be the ID of the specific Destination whose keys you want to rotate, 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 Worker Group named `default`.

1. Send a `PUT` request to the `/system/lookups` endpoint to upload the file. Example:

Single-instance Distributed

```
curl -X PUT "<url>/api/v1/m/{workerGroup}/system/lookups?filename=example.csv"
-H "Authorization: Bearer <token>" -H 'Content-Type: text/csv' \
--data-binary '@/path/to/your/example.csv'
```

You will receive a JSON object response similar to the following example:

```
{"filename":"example.csv.random.tmp","rows":100,"size":200}
```

2. Send a `POST` request referencing the Lookup file to the `/system/lookups` endpoint. Replace the filename with the response from the previous `PUT` request. This both creates the Lookup and moves the Lookup file to its final destination. Example:

Single-instance Distributed

```
curl -X POST "<url>/api/v1/m/{workerGroup}/system/lookups" \
-H "accept: application/json" -H "Authorization: Bearer <token>" \
-H "Content-Type: application/json" \
-d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
```

Example response:

```
{
  "items": [
    {
      "id": "example.csv",
      "size": 200
    }
  ],
  "count": 1
}
```

3. If replacing an existing lookup, send a `PATCH` request referencing the existing filename in the URL and the body. Example:

Single-instance Distributed

```
curl -X PATCH "<url>/api/v1/m/{workerGroup}/system/lookups/example.csv" \
-H "accept: application/json" -H "Authorization: Bearer <token>" \
-H "Content-Type: application/json" \
-d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
```

Example response:

```
{
  "items": [
    {
      "id": "example.csv",
      "size": 200
    }
  ],
  "count": 1
}
```

Distributed Upload

In a distributed environment, for lookup files of manageable size: Cribl recommends uploading the Lookup file only to the Leader Node, and then making a selective commit and deploy with only those Lookup file changes. The Leader then notifies Worker Nodes that a new configuration is available, and Worker 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 Stream 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 .

19.2. Cribl Expression Syntax

As data travels through a Cribl Stream Pipeline, it is operated on by a series of Functions. Functions are fundamentally JavaScript code.

Functions that ship with Cribl Stream are configurable via a set of inputs. Some of these configuration options are literals, such as field names, and others can be JavaScript expressions.

Expressions are **valid units** of code that resolve to a value. Every syntactically valid expression resolves to some value, but conceptually, there are two types of expressions: those that **assign** value to a variable (a.k.a., with side effects), and those that **evaluate** to a value.

Assigning a value	Evaluating to a value

`x = 42`
`newFoo = foo.slice(30)`
	`(Math.random() * 42)`
	`3 + 4`
	`'foobar'`
	`'42'`

Filters and Value Expressions

Let's consider evaluation expressions before assignment expressions.

Filters

Filters are used in Routes to select a stream of the data flow, and in Functions to scope or narrow down the applicability of a Function. Filters are expressions that **must** evaluate to either `true` (or truthy) or `false` (or falsy). Keep this in mind when creating Routes or Functions. For example:

- `sourcetype=='access_combined' && host.startsWith('web')`
- `source.endsWith('.log') || sourcetype=='aws:cloudwatchlogs:vpcflow'`

This table shows examples of truthy and falsy values.

Truthy	Falsy

`true`
`42`
`-42`
`3.14`
`"foo"`
	`false`
	`null`
	`undefined`
	`0`
	`NaN`

Truthy	Falsy
`Infinity` `-Infinity`	`''` `""`

Value Expressions

Value expressions are typically used in Functions to assign a value – for example, to a new field. For example:

- `Math.floor(_time/3600)`
- `source.replace(/.{3}/, 'XXX')`

Best Practices for Creating Predictable Expressions

- In a value expression, ensure that the source variable is not `null`, `undefined`, or `empty`.
  For example, assume you want to have a field called `len`, to be assigned the length of a second field called `employeeID`. But you're not sure if `employeeID` exists. Instead of `employeeID.length`, you can use a safer shorthand, such as: `(employeeID || '').length`.
- If a field does not exist (undefined), and you're doing a comparison with its properties, then the boolean expression will **always** evaluate to false. For example, if `employeeID` is undefined, then both of these expressions will evaluate to false: `employeeID.length > 10` , and `employeeID.length < 10` .
- `==` means "equal to," while `===` means "equal value **and** equal type." For example, `5 == 5` and `5 == "5"` each evaluate to **true**, while `5 === "5"` evaluates to **false**.
- A ternary operator is a very powerful way to create conditional values. For example, if you wanted to assign either `minor` or `adult` to a field `groupAge`, based on the value of `age`, you could do: `(age >= 18) ? 'adult' : 'minor'`.

Fields with Non-Alphanumeric Characters

If there are fields whose names include non-alphanumeric characters – e.g., `@timestamp` or `user-agent` or `kubernetes.namespace_name` – you can access them using `__e['<field-name-here>']`. (Note the single quotes.) More details here.

In any other place where the field is referenced – e.g., in the Eval function's field names – you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Wildcard Lists

Wildcard Lists are used throughout the product, especially in various Functions, such as Eval, Mask, Publish Metrics, Parser, etc.

Wildcard Lists, as their name implies, accept strings with asterisks (`*`) to represent one or more terms. They also accept strings that start with an exclamation mark (`!`) to **negate** one or more terms. This allows for implementing any combination of allowlists and blocklists.

Wildcard Lists are order-sensitive, evaluated from left to right. This is especially relevant when you use negated terms. For negations to take precedence over wildcards when evaluated, you must list negations before wildcards.

Some examples:

Wildcard List	Value	Meaning
List 1	`!foobar, foo*`	All terms that start with **foo**, except **foobar**.
List 2	`!foo*, *`	All terms, except for those that start with **foo**.
List 3	`*, !foo`	All terms (wildcard matches first, negation isn't evaluated).

You cannot use wildcards to target Cribl Stream internal fields that start with `__` (double underscore). You must specify these fields individually. For example, `__foobartab` cannot be removed by specifying `__foo*`.

19.3. Cribl Expressions

Native Cribl Stream methods can be found under `C.*`, and can be invoked from any Function that allows for expression evaluations. For example, to create a field that is the SHA1 of a another field's value, you can use the Eval Function with this **Evaluate Fields** pair:

Name	Value Expression
myNewField	`C.Mask.sha1(myOtherField)`

> Where fields' names contain special characters, you can reference them using the `__e['<field-name-here>']` convention. For details, see Fields with Non-Alphanumeric Characters.

Cribl expressions offer methods and properties in the following classes:

- C.Crypto - data encryption and decryption
- C.Encode and C.Decode - encoding and decoding data
- C.Lookup - running lookups
- C.Mask - masking sensitive data
- C.Net - network methods
- C.Text - text manipulation
- C.Time - time and date manipulation
- C.Schema - schema validation
- C.Secret - secret management
- C.env - environment properties
- C.os - system methods
- C.vars - global variables
- C.version - Cribl Stream versions
- C.Misc - various other methods

19.3.1. C.Crypto – Data Encryption And Decryption

C.Crypto.decrypt()

```
Crypto.decrypt(value: string, escape: boolean = false, escapeSeq = '"'): string
```

Decrypts all occurrences of ciphers in the given value. Instances that cannot be decrypted (for any reason) are left intact.

Returns `value` with ciphers decrypted.

Parameter	Type	Description
`value`	string	String in which to look for ciphers.
`escape`	boolean	Set to `true` to escape double quotes in output after decryption (for example, for data encrypted in Splunk.) Defaults to `false`.
`escapeSeq`	string	String used to escape double quotes. The default `'"'` escapes CSV output.

Examples

To decrypt the contents of the `encrypted` field, use:

```
C.Crypto.decrypt(encrypted)
```

C.Crypto.encrypt()

```
Crypto.encrypt(value: Buffer | string, keyclass: number, keyId?: string, defaultVal
```

Encrypts the given value with the `keyId`, or with a `keyId` picked up automatically based on `keyclass`.

Returns the encrypted value. If encryption does not succeed, returns `defaultVal` if specified; otherwise, `value`.

Parameter	Type	Description
`value`	string | Buffer	The value to encrypt.
`keyclass`	number	Key Class to pick a key from, if `keyId` isn't specified.
`keyId`	string	Encryption `[keyId``](securing-data-encryption#encryption-keys), takes precedence over keyclass`.
`defaultVal`	string	Value to return if encryption fails; if unspecified, the original value is returned.

Examples

To encrypt the `customer` field using key with `keyId` 2, use:

```
C.Crypto.encrypt(customer, -1, 2)
```

To encrypt the same field with a key selected from the Key Class 3, without specifying a `keyId`:

```
C.Crypto.encrypt(customer, 3)
```

C.Crypto.createHmac()

```
Crypto.createHmac(value: string | Buffer, secret: string, algorithm: string = 'sha2
```

Generates an HMAC that can be added to events, or can be used to validate events that contain an HMAC.

Returns the calculated HMAC digest on success; otherwise, `value`.

Parameter	Type	Description
`value`	string | Buffer	The data to encrypt. When the `outputFormat` is invalid or undefined, this parameter is returned as the digest, via a Buffer.
`secret`	string	The secret key used to generate the MAC.
`algorithm`	string	The hash algorithm used to generate the MAC. Defaults to `'sha256'`. Run `openssl list -digest-algorithms` to see

Parameter	Type	Description
		the list of available algorithms.
`outputFormat`	`'base64'` | `'hex'` | `'latin1'`	Output format for the MAC. Defaults to `'hex'`.

Examples

To generate HMAC from the `customer` field, provide the secret as the second argument:

```
C.Crypto.createHmac(customer, 'kETvYtLKZkgIzBvfRdVcZULBITANjOMd')
```

To additionally specify that you want to use the SHA512 hash algorithm and to output the HMAC in `latin1` format, run:

```
C.Crypto.createHmac(customer, 'kETvYtLKZkgIzBvfRdVcZULBITANjOMd', 'sha512', 'latin
```

19.3.2. C.Decode And C.Encode - Encoding And Decoding

C.Decode – Data Decoding Methods

`C.Decode.base64()`

```
Decode.base64(val: string, resultEnc: string = 'utf8'): string | Buffer | undefine
```

Performs base64 decoding of the given string.

Returns a string or Buffer, depending on the `resultEnc` value.

Parameter	Type	Description
`val`	string	Value to decode.
`resultEnc`	`'utf8' | 'utf8-valid' | 'buffer'`	Encoding to use to convert the binary data to a string. Use `'utf8-valid'` to validate that result is valid UTF8; use `'buffer'` if you need the binary data in a Buffer. Defaults to `'utf8'`.

Examples

To decode an `encoded` field and return the decoded value as a Buffer, use:

```
C.Decode.base64(encoded, 'buffer')
```

`C.Decode.gzip()`

```
Decode.gzip(value: Buffer | string, encoding?: BufferEncoding) : string
```

Gunzips the supplied value.

Parameter	Type	Description
`value`	any	Value to gunzip.
`encoding`	string	Encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`. Default is `'base64'`. If data is received as Buffer (from gzip with encoding: `'none'`), decoding is skipped.

Examples

To decode a `hostname` field which had been encoded using `hex`, run:

```
C.Decode.gzip(hostname, 'hex')
```

`C.Decode.hex()`

```
Decode.hex(val: string): number
```

Performs hex to number conversion. Returns `NaN` if value cannot be converted to a number.

Parameter	Type	Description
`val`	string	Hex string to parse to a number (for example, “0xcafe”).

`C.Decode.uri()`

```
Decode.uri(val: string): string
```

Performs URI-decoding of the given string.

Parameter	Type	Description
`val`	string	Value to decode.

`C.Decode.inflate()`

```
Decode.inflate(value: Buffer | string, encoding: BufferEncoding = 'base64', isRaw =
```

Inflates the supplied value.

Parameter	Type	Description
`value`	string | Buffer	The value to inflate. May be a Buffer or a string.
`encoding`	string	If a string `value` is provided, this specifies how it has been encoded. If no `encoding` is provided, `base64` is used. This parameter is ignored if `value` is already a Buffer.
`isRaw`	boolean	Indicates whether the data is in raw deflate format (without zlib headers). Default value is `false`.

Examples

To base64-decode and inflate the value contained in a `_raw` field, you can use:

```
C.Decode.inflate(_raw)
```

To additionally specify a different encoding to use, run:

```
C.Decode.inflate(_raw, 'hex')
```

C.Encode – Data Encoding Methods

`C.Encode.base64()`

```
Encode.base64(val: string | Buffer, trimTrailEq: boolean = false, encoding?: Buffe
```

Returns a base64 representation of the given string or Buffer.

Parameter	Type	Description
`val`	any	Value to encode.
`trimTrailEq`	boolean	Whether to trim any trailing `=`.
`encoding`	string	Optional parameter that specifies the encoding of the value. Options

Parameter	Type	Description
		include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples

To base64-encode the `username` field, and remove the trailing `=`, use:

```
C.Encode.base64(username, true)
```

To base64-encode the `hexValue` field, where the value of the field is a hex string:

```
C.Encode.base64('4672616e6b7575757575', undefined, 'hex')
```

`C.Encode.gzip()`

`Encode.gzip(value: Buffer | string, encoding?: string) : string | Buffer`

Gzips, and optionally base64-encodes, the supplied value.

Parameter	Type	Description
`value`	string	Value to gzip.
`encoding`	string	Encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`, `'none'`. Default is `'base64'`. If `'none'` is specified, data will be returned as a Buffer.

Examples

To encode the `hostname` field, for example using `hex` encoding, run:

```
C.Encode.gzip(hostname, 'hex')
```

`C.Encode.hex()`

```
Encode.hex(val: string | number): string
```

Rounds the number to an integer and returns its hex representation (lowercase). If a string is provided, it will be parsed into a number or `NaN`.

Parameter	Type	Description
`val`	string | number	Value to convert to hex.

`C.Encode.uri()`

```
Encode.uri(val: string): string
```

Returns the URI-encoded representation of the given string.

Parameter	Type	Description
`val`	string	Value to encode.

`C.Encode.deflate()`

```
Encode.deflate(value: Buffer | string, encoding: BufferEncoding | 'none' = 'base64
```

Deflates and optionally base64-encodes the supplied value.

Parameter	Type	Description
`value`	string | Buffer	Value to deflate.
`encoding`	string	Encoding to apply to the deflated result, for example: `base64`, `hex`, `utf-8`, `binary`, `none`. Default is `base64`. If `none` is specified, data will be returned as a Buffer.
`toRaw`	boolean	Indicates whether to return the raw deflated data, without zlib headers. Default is `false`, which will prepend normal zlib header bytes to the returned value.

Examples

To deflate and hex-decode the value provided in an `inflated` field, run:

```
C.Encode.deflate(inflated, 'hex')
```

19.3.3. C.Lookup – Inline Lookup Methods

`C.Lookup` – Exact Lookup

```
Lookup: (file: string, primaryKey?: string, otherFields: string[]=[],ignoreCase: bo
```

Returns an instance of a lookup to use inline.

Examples

In this example, `host` is the name of the primary key field:

```
C.Lookup('lookup_name.csv', 'IP_field_name_in_lookup_file').match(host)
```

Here, the quoted `'event_field_or_string_to_match'` could be a string to match in the primary key field:

```
C.Lookup('name_of_lookup_file.csv', 'field_in_csv_to_match').match('event_field_or_
```

This example checks whether `someValue` is present or not and returns the answer as a boolean.

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue')
```

This expression returns a string – the `fieldToReturn` column from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol', additionalCols).match('someValue','fieldToReturn
```

This expression returns an array – the specified `fieldToReturn1` and `fieldToReturn2` columns from the matched lookup row:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', ['fieldToReturn1', 'fieldToRe
```

To return an array with all columns from the matched lookup row, you can use:

```
C.Lookup('file', 'primaryKeyCol').match('someValue', [])
```

`C.Lookup` can load lookup files of up to 10 MB.

All inputs to Lookup methods' `match()` method must be strings. If your lookup file contains numeric fields, convert them to strings, e.g.: `.match(String(<fieldname>)`.

You can use the optional `otherFields[]` argument, shown in the above `C.Lookup()` signatures and examples, to limit which columns of the lookup table will be available in a subsequent `.match()` call. If omitted, or set to `undefined`, all columns will be available.

Cribl Stream 4.1 and later support returning all columns from a matched row. Use the empty-array convention shown in the above examples: `.match('someValue',[])`.

`C.LookupCIDR` – CIDR Lookup

```
LookupCIDR: (file: string, primaryKey?: string, otherFields: string[]=[]) => Inline
```

Returns an instance of a CIDR lookup to use inline.

`C.LookupIgnoreCase` – Case-insensitive Lookup

```
LookupIgnoreCase: (file: string, primaryKey?: string, otherFields: string[]=[]) =>
```

Returns an instance of a lookup (ignoring case) to use inline. Works identically to `C.Lookup`, except ignores the case of lookup values. (Equivalent to calling `C.Lookup` with its fourth `ignoreCase?` parameter set to `true`).

`C.LookupRegex` – Regex Lookup

```
LookupRegex: (file: string, primaryKey?: string, otherFields: string[]=[]) => Inli
```

Returns an instance of a Regex lookup to use inline.

`C.Lookup.match()`

```
InlineLookup.match(value: string): boolean

InlineLookup.match(value: string, fieldToReturn?: string): any
InlineLookup.match(value: string, fieldToReturn: string[]): {[key: string]: any}
```

Parameter	Type	Description
`value`	string	the value to look up.
`fieldToReturn`	string | string []	name of the lookup file > field to return.

Examples

To find the last row where the value from column `foo` matches the string `abc`, you can use the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')
```

To find the last row where the value from column `transport` matches the string `tcp`, you can use the following expression. If a matching row is found, the expression returns the row's value for column `port_number`.

```
C.Lookup('service_names_port_numbers.csv', 'transport').match('tcp', 'port_number'
```

You can find the last row where the CIDR range in `cidr` includes `192.168.1.1` with the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.LookupCIDR('lookup-cidr.csv', 'cidr').match('192.168.1.1', 'bar')
```

The following expression finds the last row where the value from column `cidr` matches `hostIP`. If a matching row is found, it returns the row's value for column `location`.

```
C.LookupCIDR('lookup-cidr.csv', 'cidr').match(hostIP, 'location')
```

To find the last row where the regex in column `foo` matches the string `manchester`, use the following expression. If a matching row is found, the expression returns the row's value for column `bar`.

```
C.LookupRegex('lookup-regex.csv', 'foo').match('manchester', 'bar')
```

With `C.LookupRegex`, ensure that your lookup file contains no empty lines – not even at the bottom. Any empty rows will cause `C.LookupRegex().match()` to always return `true`.

19.3.4. C.Mask – Data Masking Methods

C.Mask.CC()

```
Mask.CC(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks whether a value could be a valid credit card number, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** it could be a valid credit card number.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

Examples

Let's assume your parsed data contains a `creditCard` field. You can mask this number by using a Mask Function. Apply it to the `cardNumber` field and create the following masking rule:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.CC(g1)}`	`cardNumber: XXXXXXXXXX4860`

To control how the masking is performed, you can specify how many digit should be unmasked and what masking character to use:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.CC(g1, 4, "Z")}`	`cardNumber: 6011ZZZZZZZZZZZZ`

C.Mask.IMEI()

```
Mask.IMEI(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks whether a value could be a valid IMEI number, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`.

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** it could be a valid IMEI number.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

Examples

Let's assume your parsed data contains an `imei` field. You can mask this number by using a Mask Function. Apply it to the `imei` field and create the following masking rule:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.IMEI(g1)}`	`imei: XXXXXXXXXXX7011`

To control how the masking is performed, you can specify how many digit should be unmasked and what masking character to use:

Match Regex	Replace Expression	Example result
`(.*)` (catches the whole field contents)	`${C.Mask.CC(g1, 4, "M")}`	`cardNumber: 5356MMMMMMMMMMMM`

C.Mask.isCC()

```
Mask.isCC(value: string): boolean
```

Checks whether the given value could be a valid credit card number, by computing the string's Luhn's checksum modulo 10 == `0`.

Parameter	Type	Description
`value`	string	String to check for being a valid credit card number.

Examples

The following example expression uses `isCC()` to check whether the `cardNumber` field actually contains a credit card number. If it does, the card number is replaced with the `REDACTED` string, otherwise, en empty string is returned.

```
C.Mask.isCC(cardNumber) ? C.Mask.REDACTED : ""
```

C.Mask.isIMEI()

```
Mask.isIMEI(value: string): boolean
```

Checks whether the given value could be a valid IMEI number, by computing the string's Luhn's checksum modulo 10 == `0`.

Parameter	Type	Description
`value`	string	String to check for being a valid IMEI number.

Examples

The following example expression uses `isCC()` to check whether the `imei` field actually contains ab IMEI number number. If it does, the IMEI number is replaced with the `REDACTED` string, otherwise, en empty string is returned.

```
C.Mask.isIMEI(imei) ? C.Mask.REDACTED : ""
```

C.Mask.luhn()

```
Mask.luhn(value: string, unmasked: number = -4, maskChar: string='X'): string
```

Checks that value Luhn's checksum mod 10 is `0`, and masks a subset of the value. By default, all digits except the last 4 will be replaced with `X`. If the value's Luhn's checksum mod 10 is not `0`, then the value is returned unmodified.

Parameter	Type	Description
`value`	string	The string to mask, **if and only if** the value's Luhn's checksum mod 10 is `0`.
`unmasked`	number	Number of digits to leave unmasked: positive for left, negative for right, `0` for none.
`maskChar`	string	String to replace each digit with.

C.Mask.luhnChecksum()

```
Mask.luhnChecksum(value: string, mod: number=10): number
```

Generates the Luhn checksum (used to validate certain credit card numbers, IMEIs, etc.). By default, the mod 10 of the checksum is returned. Pass mod = `0` to get the actual checksum.

Parameter	Type	Description
`value`	string	String whose digits you want to perform the Luhn checksum on.
`mod`	number	Return checksum modulo this number. If `0`, skip modulo. Default is `10`.

C.Mask.md5()

```
Mask.md5(value: string, len?: string | number, encoding?: BufferEncoding): string
```

Generates MD5 hash of a given value.

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string | number	Length of hash to return: 0 for full hash, a +number for left or a -number for right substring. If a string is passed it's length will be used.

Parameter	Type	Description
`encoding`	string	Optional parameter that specifies the encoding of the value. Options include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples

To hash the contents of a `username` field with the MD5 algorithm, run:

```
C.Mask.md5(username)
```

To hash the contents of an `authToken` field, where the value of the field is a base64-encoded string, with the MD5 algorithm, run:

```
C.Mask.md5(authToken, undefined, 'base64')
```

`C.Mask.random`

```
Mask.random(len?: string | number): string
```

Generates a random alphanumeric string.

Parameter	Type	Description
`len`	string | number	Length of the result; or, if a string, use its length.

Examples

Assuming you have some identifying user information in a `username` field, you can replace it with a random string of the same length with:

```
C.Mask.random(username)
```

Input	Output
`McGoatFace`	“xY2KLHkyTg”

`C.Mask.repeat()`

```
Mask.repeat(len?: number | string, char: string = 'X'): string
```

Generates a repeating char/string pattern, e.g., `XXXX`.

Parameter	Type	Description
`len`	string | number	Length of the result; or, if a string, use its length.
`char`	string	Pattern to repeat `len` times.

Examples

The following example will repeat the "GOAT" string as many time as there are characters in the first parameter, "goat", so four times:

```
C.Mask.repeat("goat", "GOAT")
```

`C.Mask.sha1()`, `C.Mask.sha256()`, `C.Mask.sha512()`

```
Mask.sha1(value: string, len?: string | number, encoding?: BufferEncoding): string
Mask.sha256(value: string, len?: string | number, encoding?: BufferEncoding): strin
Mask.sha512(value: string, len?: string | number, encoding?: BufferEncoding): strin
```

Generates SHA1, SHA256, or SHA512 hash of given value.

Parameter	Type	Description
`value`	string	The string to hash.
`len`	string | number	Length of hash to return: `0` for full hash, a +number for left, or a -number for right substring. If a string is passed, its length will be used.
`encoding`	string	Optional parameter that specifies the encoding of the value. Options include `base64`, `hex`, `utf-8`, and `binary`. The default is `utf-8`.

Examples

To hash the contents of a `secret` field with the SHA256 algorithm, run:

```
C.Mask.sha256(secret)
```

To hash the contents of an `authToken` field, where the value of the field is a base64-encoded string, with the SHA256 algorithm, run:

```
C.Mask.sha256(authToken, undefined, 'base64')
```

`C.Mask.REDACTED`

```
Mask.REDACTED: string
```

The literal `'REDACTED'`.

19.3.5. C.Net – Network Methods

`C.Net.cidrMatch()`

```
Net.cidrMatch(cidrIpRange: string, ipAddress: string): boolean
```

Determines whether the supplied IPv4 or IPv6 `ipAddress` is inside the range of addresses identified by `cidrIpRange`.

Parameter	Type	Description
`cidrIpRange`	string	IP address range in CIDR format. E.g., `10.0.0.0/24`.
`ipAddress`	string	The IP address to test for inclusion in `cidrIpRange`.

Examples

(IPv4): `C.Net.cidrMatch('10.0.0.0/24', '10.0.0.100')` returns `true`.

(IPv6): `C.Net.cidrMatch('2001:db8::/32', '2001:db8:0:0:0:ffff:0:0')` returns `true`.

`C.Net.isIpV4()`

```
Net.isIpV4(value: string): boolean
```

Determines if the value supplied is an IPv4 address.

Parameter	Type	Description
`value`	string	The IP address to test.

`C.Net.isIpV6()`

```
Net.isIpV6(value: string): boolean
```

Determines if the value supplied is an IPv6 address.

Parameter	Type	Description
`value`	string	The IP address to test.

C.Net.isIPV4Cidr()

```
Net.isIPV4Cidr(value: string): boolean
```

Determines whether a value supplied is an IPv4 CIDR range.

Returns `true` if the value is a valid IPv4 CIDR range; otherwise, `false`.

Parameter	Type	Description
`value`	string	String value that may be an IPv4 CIDR range.

C.Net.isIPV6Cidr()

```
Net.isIPV6Cidr(value: string): boolean
```

Determine whether a value supplied is an IPv6 CIDR range.

Returns `true` if the value is a valid IPv6 CIDR range; otherwise, `false`.

Parameter	Type	Description
`value`	string	String value that may be an IPv6 CIDR range.

C.Net.isIpV4AllInterfaces()

```
Net.isIpV4AllInterfaces(value): boolean
```

Determines if the value supplied is the IPv4 all-interfaces address, typically used to bind to all IPv4 interfaces – i.e., ‘0.0.0.0’.

Parameter	Type	Description
`value`	any	The IP address to test.

C.Net.isIpV6AllInterfaces()

```
Net.isIpV6AllInterfaces(value): boolean
```

Determines 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'.

Parameter	Type	Description
`value`	any	the IP address to test.

C.Net.ipv6Normalize()

```
Net.ipv6Normalize(address: string): string
```

Normalizes an IPV6 address based on RFC draft-ietf-6man-text-addr-representation-04.

Parameter	Type	Description
`address`	string	The IPV6 address to normalize.

C.Net.isPrivate()

```
Net.isPrivate(address: string): boolean
```

Determines whether the supplied IPv4 address is in the range of private addresses per RFC1918.

Parameter	Type	Description
`address`	string	The IP address to test.

19.3.6. C.Text – Text Methods

`C.Text.entropy()`

```
Text.entropy(bytes: Buffer | string): number
```

Computes the Shannon entropy of the given Buffer or string.

Returns the entropy value; or `-1` in case of an error.

Parameter	Type	Description
`bytes`	Buffer | string	value to undergo Shannon entropy computation.

`C.Text.hashCode()`

```
Text.hashCode(val: string | Buffer | number): number
```

Computes hashcode (djb2) of the given value.

Returns hashcode value.

Parameter	Type	Description
`val`	string | Buffer | number	value to be hashed.

`C.Text.isASCII()`

```
Text.isASCII(bytes: Buffer | string): boolean
```

Checks whether all bytes or chars are in the ASCII printable range.

Returns `true` if all chars/bytes are within ASCII printable range; otherwise, `false`.

Parameter	Type	Description
`bytes`	string | Buffer	value to check for character range.

C.Text.isUTF8()

```
Text.isUTF8(bytes: Buffer | string): boolean
```

Checks whether the given Buffer contains valid UTF8.

Returns `true` if bytes are UTF8; otherwise, `false`.

Parameter	Type	Description
`bytes`	Buffer | string	bytes to check.

C.Text.parseWinEvent()

```
Text.parseWinEvent(xml: string, nonValues: string[] = Text._WIN_EVENT_NON_VALUES):
```

Parses an XML string representing a Windows event into a compact, prettified JSON object. Works like `C.Text.parseXml`, but with Windows events, produces more-compact output. For a usage example, see Reducing Windows XML Events.

Returns an object representing the parsed Windows Event; or `undefined` if the input could not be parsed.

Parameter	Type	Description
`xml`	string	an XML string; or an event field containing the XML.
`nonValues`	string[]	array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `['-']`, meaning that elements whose value equals `-` will be discarded.

C.Text.parseXml()

```
Text.parseXml(xml: string, keepAttr: boolean = true, keepMetadata: boolean = false
```

Parses an XML string and returns a JSON object. Can be used with Eval Function to parse XML fields contained in an event, or with ad hoc XML.

Returns an object representing the parsed XML; or `undefined` if the input could not be parsed. An input collection of elements will be parsed into an array of objects.

Parameter	Type	Description
`xml`	string	XML string, or an event field containing the XML.
`keepAttr`	boolean	whether or not to include attributes in the returned object. Defaults to `true`.
`keepMetadata`	boolean	whether or not to include metadata found in the XML. The `keepAttr` parameter must be set to `true` for this to work. Defaults to `false`. (Eligible metadata includes namespace definitions and prefixes, and XML declaration attributes such as encoding, version, etc.)
`nonValues`	string[]	array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `[]` (empty array), meaning discard no elements.

C.Text.relativeEntropy()

```
Text.relativeEntropy(bytes: Buffer | string, modelName: string = 'top_domains'): nu
```

Computes the relative entropy of the given Buffer or string.

Returns the relative entropy value, or `-1` in case of an error.

Parameter	Type	Description
`bytes`	Buffer | string	Value whose relative entropy to compute.
`modelName`	string	Optionally, override the default `$CRIBL_HOME/data/lookups/model_relative_entropy_top_domains.csv` model used to test the input. Create a custom lookup file with the same column and value structure as the default, and store it in the same path, as `model_relative_entropy_<custom-name>.csv`. To reference it, pass your `<custom-name>` substring as the `modelName` parameter.

When using `modelName` in a distributed deployment, the corresponding paths are `$CRIBL_HOME/groups/<worker-group-name>/data/lookups/`. Creating your custom lookup file via Cribl Stream's UI will automatically set the appropriate paths.

19.3.7. C.Time – Time Methods

C.Time.adjustTZ()

```
Time.adjustTZ(epochTime: number, tzTo: string, tzFrom: string = 'utc'): number | ui
```

Adjusts a timestamp from one timezone to another.

Returns the adjusted timestamp, in UNIX epoch time (ms).

Parameter	Type	Description
`epochTime`	number	Timestamp to adjust, in UNIX epoch time.
`tzTo`	string	Timezone to adjust to (full name).
`tzFrom` (optional)	string	Timezone of the timestamp.

Cribl relies on this list of TZ Database Time Zones.

Note that these are timezone identifiers (full names), not abbreviations such as “CET”, “PST”, “EST”, and so on.

Examples

To adjust a timestamp found in the `_time` field to a selected timezone (here, Australia/Melbourne), use:

```
C.Time.adjustTZ(_time, "Australia/Melbourne")
```

The method returns the timestamp in milliseconds. If you want to further operate on it using methods such as `strftime()`, which takes as parameter UNIX epoch time in seconds, you can divide it by 1000:

```
C.Time.strftime(C.Time.adjustTZ(_time, "Australia/Melbourne") / 1000, "%d %B %Y %H
```

C.Time.clamp()

```
Time.clamp(date: T, earliest: T, latest: T, defaultDate?: T ): T  | undefined
```

Constrains a parsed timestamp to realistic earliest and latest boundaries.

Returns a JavaScript Date object or timestamp, depending on the format of the `date` parameter.

Parameter	Type	Description
`date`	Date | number	Timestamp to constrain, in UNIX epoch time (ms) or JavaScript Date format.
`earliest`	Date | number	Earliest allowable timestamp, in UNIX epoch time (ms) or JavaScript Date format.
`latest`	Date | number	Latest allowable timestamp, in UNIX epoch time (ms) or JavaScript Date format.
`defaultDate` (optional)	Date | number	Default date, in UNIX epoch time (ms) or JavaScript Date format. This date is used for values that fall outside the `earliest` or `latest` boundaries.

Examples

Suppose you have a date in a "YYYY-MM-DD" format in a `time` field. To make sure this date does not exceed the range of 1 Jan 2020 – 1 Jan 2025, you can use:

```
C.Time.clamp(time, "2020-01-01", "2025-01-01")
```

Input	Output
"2016-02-11"	"2020-01-01" (set to the earliest value)
"2026-01-01"	"2025-01-01" (set to the latest value)
"2023-12-24"	"2023-12-24" (unchanged, because it fits inside the defined range)

To make sure that all dates that fall outside the bounds are changed to a specific value, add the third parameter:

```
C.Time.clamp(time, "2020-01-01", "2025-01-01", "2022-02-22")
```

Input	Output
“2016-02-11”	“2022-02-22” (set to the default value)
“2026-01-01”	“2022-02-22” (set to the default value)
“2023-12-24”	“2023-12-24” (unchanged, because it fits inside the defined range)

`C.Time.strftime()`

```
Time.strftime(date: number | Date | string, format: string, utc: boolean = true):
```

Formats a Date object or timestamp number as a time string, using d3js time format.

Returns representation of the given date in the specified format.

Parameter	Type	Description
`date`	number | Date | string	Date to format, in UNIX epoch time (in seconds) or JavaScript Date format.
`format`	string	New format for the date.
`utc` (optional)	boolean	Whether to output the time in UTC (`true`), rather than in local timezone.

Examples

To transform a `_time` field containing a UNIX timestamp to a string with time formatted, for example: `15 November 2023, 08:03`, you can use:

```
C.Time.strftime(_time, "%d %B %Y, %H:%M")
```

Some common formats (assuming UNIX timestamp of 1702460134) are:

Format	Output
“%Y-%m-%d” (ISO date)	“2023-12-13”
“%d %B %Y %H:%M:%S”	“13 December 2023 09:35:34”

Format	Output
“%x %X” (the locale’s date and time format)	“12/13/2023 9:35:34 AM”

For detailed reference of the date and time tokens, see d3js reference or Auto Timestamp.

`C.Time.strptime()`

```
Time.strptime(str: string, format: string, utc: boolean = true, strict: boolean =
```

Extracts time from a string using d3js time format that defines the format of the incoming string.

Returns a parsed JavaScript Date object, or `null` if the specified `format` did not match.

Parameter	Type	Description
`str`	string	Timestamp to parse.
`format`	string	Format to parse.
`utc` (optional)	boolean	Whether to interpret times as UTC (`true`), rather than as local time.
`strict` (optional)	boolean	Whether to return `null` if there are any extra characters after timestamp.

Examples

To convert a string in the “YYYY-MM-DD” format, for example “2023-12-12”, to a Date object, use:

```
C.Time.strptime(F, "%Y %m %d")
```

Additionally, to interpret the time in UTC, and make sure null is returned if the timestamp contains any trailing characters, use:

```
C.Time.strptime(F, "%Y %m %d", true, true)
```

For detailed reference of the date and time tokens, see d3js reference or Auto Timestamp.

`C.Time.timestampFinder()`

```
Time.timestampFinder(utc?: boolean).find(str: string): IAutoTimeParser
```

Extracts time from the specified field, using the same algorithm as the Auto Timestamp Function and the Event Breaker Function.

Returns representation of the extracted time; truncates timestamps to three-digit (milliseconds) resolution, omitting trailing zeros.

Parameter	Type	Description
`utc`	boolean	Whether to output the time in UTC (`true`), rather than in local timezone.
`str`	string	The field in which to search for the time.

Examples

To find timestamps in the `_raw` field and output them in UTC, you can use:

```
C.Time.timestampFinder(true).find(_raw)
```

You can then format the string output with `strftime()`:

```
C.Time.strftime(C.Time.timestampFinder(true).find(_raw), "%d.%m.%Y, %H.%M")
```

19.3.8. Miscellaneous Expression Methods

C.Schema – Schema Methods

`C.Schema`

```
Schema: (id: string) => SchemaValidator
```

`C.Schema.validate()`

```
SchemaValidator.validate(data: any): boolean
```

Validates the given object against the schema.

Returns `true` when schema is valid; otherwise, `false`.

Parameter	Type	Description
`data`	any	Object to be validated.

Examples

To validate whether a `myField` conforms to `schema1`, you can use:

```
C.Schema('schema1').validate(myField)
```

See Schema Library for more details.

C.Secret – Secrets-Management Methods

`C.Secret()`

```
Secret: (id: string, type?: string): ISecret
Secret(id: string, type: 'keypair') => IPairSecret
Secret(id: string, type: 'text') => ITextSecret
Secret(id: string, type: 'credentials') => ICredentialsSecret
```

Returns a secret matching the specified ID.

Parameter	Type	Description
`id`	string	ID of the secret.
`type` (optional)	'text' | 'keypair' | 'credentials'	Type of the secret.

Examples

To return a text secret with ID `victorias` (or with undefined, if no such secret exists), use:

```
C.Secret('victorias', 'text')
```

You can also get attributes of secrets with the following expressions:

```
C.Secret('api_key', 'keypair').secretKey
C.Secret('secret_hash', 'text').value
C.Secret('user_pass', 'credentials').password
```

Common returned attributes for `ISecret` objects:

- `secretType` – one of `keypair`, `text`, or `credentials`.
- `description` (optional) – the secret description.
- `tags` (optional) – a comma separated list of tags.

Additional returned attributes for `IPairSecret` objects:

- `apiKey` – the API key value
- `secretKey` – the Secret key value

Additional returned attributes for `ITextSecret` objects:

- `value` – the text value

Additional returned attributes for `ICredentialsSecret` objects:

- `username` – the username value
- `password` – the password value

See Securing Cribl Stream > Secrets for more details.

C.env – Environment

`C.env`

```
env: {[key: string]: string;}
```

Returns an object containing Cribl Stream's environment variables.

Examples

To return the parent of Cribl's `bin` directory (generally `/opt/cribl`), use:

```
C.env.CRIBL_HOME
```

To return the hostname of the machine where Cribl Stream is running, use:

```
C.env.HOSTNAME
```

C.os – System Methods

`C.os.hostname()`

Returns hostname of the system running this Cribl Stream instance.

C.vars – Global Variables

See Global Variables Library for more details.

C.version – Cribl Stream Versions

`C.version`

Returns the Cribl Stream version currently running.

`C.confVersion`

Returns the commit hash of the Worker Node's current config version. (Evaluates only against live data sent through Worker Nodes. Values will be undefined in the Leader's Preview pane.)

C.Misc – Miscellaneous Utility Methods

`C.Misc.zip()`

```
Misc.zip(keys: string[], values: any[], dest?: any): any
```

Sets the given keys to the corresponding values on the given `dest` object. If `dest` is not provided, a new object will be constructed.

Returns object on which the fields were set.

Parameter	Type	Description
`keys`	string[]	Field names corresponding to keys.
`values`	any[]	Values corresponding to values.
`dest`	any	Object on which to set field values.

Examples

Let's take the following expression:

```
people = C.Misc.zip(titles, names)
```

If sample data contains: `titles=['ceo', 'svp', 'vp']`, `names=['foo', 'bar', 'baz']`, this expression create an object called `people`, with key names from elements in `titles`, and with corresponding values from elements in `names`.

Result: `"people": {"ceo": "foo", "svp": "bar", "vp": "baz"}`

`C.Misc.uuidv4()`

```
C.Misc.uuidv4(): string
```

Returns a version 4 (random) UUID in accordance with RFC-4122.

Examples

To create a UUIDv4, use:

```
C.Misc.uuidv4()
```

Result: a conforming UUIDv4, such as `58d8be36-4db0-4b1c-ac80-28bb03c45e0d`. It is highly improbable that two version 4 UUIDs will ever have the same value.

`C.Misc.uuidv5()`

```
C.Misc.uuidv5(name: string, namespace: string): string
```

Returns a version 5 (namespaced) UUID in accordance with RFC-4122 for the given `name` and `namespace`. If `namespace` is not a valid UUID, this function will fail.

Parameter	Type	Description
`name`	string	Any arbitrary name to use in UUID generation.
`namespace`	string	One of `DNS`, `URL`, `OID`, or `X500` to use a predefined namespace, or else a valid UUID.

Examples

To create a UUIDv5 with the predefined DNS namespace, use:

```
C.Misc.uuidv5('example', 'DNS')
```

Result: a UUID that is highly likely to be the same for the same name and namespace. In this case, the result would be `7cb48787-6d91-5b9f-bc60-f30298ea5736`.

`C.Misc.validateUUID()`

```
C.Misc.validateUUID(maybeUUID: string): boolean
```

Returns `true` if `maybeUUID` is a valid UUID of any version, and false otherwise.

Parameter	Type	Description
`maybeUUID`	string	A string to test for a valid UUID.

Examples

```
C.Misc.validateUUID(C.Misc.uuidv4())
```

Result: returns `true`

```
C.Misc.validateUUID('clearly not a UUID')
```

Result: returns `false`

`C.Misc.getUUIDVersion()`

```
C.Misc.getUUIDVersion(uuid: string): number
```

Returns a `number` of the UUID version given by `uuid` if it is a valid UUID, otherwise `undefined`.

Parameter	Type	Description
`uuid`	string	A UUID for which to determine the version.

Examples

```
C.Misc.getUUIDVersion(C.Misc.uuidv4())
```

Result: `4`

```
C.Misc.getUUIDVersion(C.Misc.uuidv5('example', 'X500'))
```

Result: `5`

19.4. CLI Reference

Command line interface basics

In addition to starting and stopping the Cribl Stream server, Cribl Stream'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 Stream with the `CRIBL_VOLUME_DIR` variable, all subsequent CLI commands should have this variable defined. Otherwise, those commands will apply Cribl Stream'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: 4.4.2
Usage: [sub-command] [options] [args]

Commands:
help                - Display help
mode-edge           - Configure instance in Edge mode
mode-managed-edge - Configure instance in Managed-Edge mode
mode-master         - Configure instance in Leader mode
mode-single         - Configure instance in Single-Instance mode
mode-worker         - Configure instance in Worker mode
reload              - Reload instance
restart             - Restart instance
start               - Start instance
status              - Display status
stop                - Stop instance
version             - Print version

auth                - Authentication
boot-start          - Enable/Disable boot-start
diag                - Manage diagnostics bundles
git                 - Manage Worker Groups config
keys                - Manage encryption keys
nc                  - Listen on a port for traffic and output stats and data
node                - Execute a JavaScript file
pack                - Manage Cribl Packs
parquet             - Manage Parquet files and schemas
pipe                - Feed stdin to a pipeline
vars                - Manage global variables
```

Starting in version 3.0, Cribl Stream's former "master" application components have been renamed "Leader." As long as some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect the options available.

`mode-master`

Configures Cribl Stream 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
  [-k <privKeyPath>]         – Path on server to the private key to use. PEM format. Ca
  [-c <certPath>]            – Path on server to the certificate to use. PEM format. Ca
  [-u <authToken>]           – Optional authentication token to include as part of the 
  [-i <ipWhitelistRegex>]    – Regex matching IP addresses that are allowed to establis
  [-r <resiliency>]          – Resiliency mode for the Leader. Accepts agruments: none,
  [-v <failoverVolume>]      – If the -r option is set to failover, this defines the vo
```

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]
```

Options

```
[-H <host>]   - Hostname/IP (defaults to 127.0.0.1).
[-p <port>]   - Port (defaults to 9420).
[-s <socket>] - Location of the AppScope socket.
```

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 Wo
[-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 ref
[-c <certPath>]     - Path on server to the certificate to use. PEM format. Can ref
[-u <authToken>]    - Authentication token to include as part of the connection hea
[-e <envRegex>]     - Regex that selects environment variables to report to Leader.
[-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.
```

When you generate bootstrap scripts via the UI to add or update Workers, these scripts automatically set the `-S` option according to the Leader's TLS configuration.

`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 Wo
[-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 ref
[-c <certPath>]      - Path on server to the certificate to use. PEM format. Can ref
[-u <authToken>]     - Authentication token to include as part of the connection hea
[-e <envRegex>]      - Regex that selects environment variables to report to Leader.
[-t <tags>]          - Tag values to report to Leader.
[-g <group>]         - Worker Group/Fleet to report to Leader.
```

Sample Response

```
Settings updated.
```

When you generate bootstrap scripts via the UI to add or update Workers, these scripts automatically set the `-S` option according to the Leader's TLS configuration.

pack

Manages Cribl Packs.

Usage

```
./cribl pack <sub-command> <options> <args>
```

Sub-commands and Options

```
export             - Export Cribl Packs, args:
   -m <mode>       - Mode to export. Accepts: merge_safe, merge, default_only.
  [-o <filename>] - Where to export the pack on disk.
  [-n <name>]     - Name to override the installed pack's name on export.
  [-g <group>]    - The Worker Group/Fleet to execute within
install            - Install a Cribl Pack, args:
  [-d]            - Run install in debug.
  [-f]            - Force install.
  [-c]            - Disallow installation of Packs with custom functions.
  [-n <name>]     - Name of the pack to install; defaults to source.
  [-g <group>]    - The Worker Group to execute within
list               - List Cribl Packs, args:
  [-v]            - Display all pack info
  [-g <group>]    - The Worker Group to execute within
uninstall          - Uninstall a Cribl Pack, args:
  [-d]            - Run uninstall in debug
  [-g <group>]    - The Worker Group 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 to execute within
```

Sample Response

```
id          version  spec  displayName    author       description
-----------------------------------------------------------------------------------
HelloPacks  1.0.0    ----  Hello, Packs!  Cribl, Inc.  A sample pack with a simple
```

`parquet`

For any Parquet file: view (`cat`) the file, its metadata (`details`), or its `schema`. This command is available only on Linux.

Usage

```
./cribl parquet <sub-command> -f <file>
```

Sub-commands and Options

```
cat               - View the file, args:
  -f <file>       - Path to file
  [-m <max>]      - Maximum number of events to print
details           - View the file's metadata, args:
  -f <file>       - Path to file
schema            - View the file's Parquet schema, args:
  -f <file>       - Path to file
```

Examples

View a Parquet file:

```
./cribl parquet cat -f /some_parquet_file
```

```
{"__bytes":274,"Unnamed: 0":0,"name":"apples","quantity":10,"price":2.6,"date":"20
...
```

View a Parquet file’s metadata, including encoding and compression:

```
./cribl parquet details -f /some_parquet_file
```

```
{
  "schema": {
    "Unnamed: 0": {
      "optional": true,
      "type": "INT64"
    },
    ...
  },
  "path": "../../../src/sluice/js/util/__tests__/data/parquet/single/small_fruits.p
  "num_rows": 40000,
  "num_cols": 11,
  "num_real_cols": 11,
  "num_row_groups": 1,
  "created_by": "parquet-cpp-arrow version 11.0.0",
  "schema_root_name": "schema",
  "metadata": {
    "pandas": "{\"index_columns\": [{\"kind\": \"range\", \"name\": null, \"start\'
    ...
  },
  "version": "PARQUET_2_6",
  "metadataSize": 6072,
  ...
    }
  ]
}
```

View a Parquet file's schema (not including encoding or compression), expressed as JSON like in the Parquet Schema editor:

```
./cribl parquet schema -f /some_parquet_file
```

```
{
  "Unnamed: 0": {
    "optional": true,
    "type": "INT64"
  },
  "name": {
    "optional": true,
    "type": "STRING"
  },
  "quantity": {
    "optional": true,
    "type": "DOUBLE"
  },
  "price": {
    "optional": true,
    "type": "DOUBLE"
...
  }
}
```

`reload`

Reloads Cribl Stream. Executes immediately.

Usage

```
./cribl reload [--help]
```

Sample Response

```
Reload request submitted to Cribl
```

`restart`

Restarts Cribl Stream. 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 Stream. Executes immediately. Upon first run, echoes Cribl Stream'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 Stream, 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 Stream. 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 Stream version. Executes immediately.

Usage

`./cribl version [--help]`

Sample Response

```
Software Version: 3.1.0-f765e418
```

`auth`

Log into or out of Cribl Stream.

Usage

`./cribl auth <sub-command> <options> <args>`

Sub-commands and Options

```
login              - Log in to Cribl Stream/Edge, args:
  [-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 Stream 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 <maxIncludeJobs>]    - Latest number of jobs to include in bundle
  [-M]                     - Exclude metrics from bundle
  [-g]                     - Exclude git log from bundle
list                       - List existing Cribl Stream/Edge diagnostic bundles
send                       - Send Cribl Stream/Edge diagnostics bundle to Cribl Suppo
   -c <caseNumber>         - Cribl Support Case Number
  [-p <path>]              - Diagnostic bundle path (if empty then new bundle will be
heapsnapshot               - Generate heap snapshot of a Cribl Stream/Edge process, a
  [-p <pid>]               - The pid of the process to dump the heap snapshot
```

Sample Responses

`diag create`

```
Created a Cribl diagnostic bundle at /opt/cribl/diag/<product>-zedborcdb72f-202108
```

`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)
  [-c <keyclass>]  - Key class to set for the key.
  [-k <kms>]       - KMS to use. Must be configured; see cribl.yml.
  [-e <expires>]   - Expiration time, in epoch time.
  [-i]             - Use an initialization vector. (IV size configurable for algor
  [-s <ivSize>]    - (For algo 'aes-256-gcm' only) Initialization vector size (byte
  [-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 <family>]         - If this is "6" then nc will listen on ::1, otherwise it list
[-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, ar
```

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
[-t]               - Perform pipeline tracing
[-a <pack>]        - Optional Cribl Pack context. Mutually exclusive with -b.
[-b <project>]     - Optional Cribl Project context
[-i]               - Runs the pipe command without sandboxing javascript expression
```

Sample Response

```
...
{"time":"2021-08-20T20:37:00.017Z","cid":"api","channel":"commands","level":"info"
{"time":"2021-08-20T20:37:00.019Z","cid":"api","channel":"pipe:main","level":"info
{"time":"2021-08-20T20:37:00.021Z","cid":"api","channel":"pipe:main","level":"info
{"time":"2021-08-20T20:37:00.022Z","cid":"api","channel":"commands","level":"info"
{"time":"2021-08-20T20:37:00.028Z","cid":"api","channel":"GrokMgr","level":"info",
...
```

`scope`

Greps your apps by the syscalls. Executes immediately.

See the AppScope CLI Reference for usage and examples.

`vars`

Manages Cribl Stream 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"
  }
]
```

19.5. Environment Variables

This is a consolidated list of environment variables available to configure Cribl Stream instances.

Distributed Deployment

You can use the following environment variables to configure your distributed Cribl Stream instance.

Name	Purpose
`CRIBL_DIST_MASTER_URL`	URL of the Leader Node. Example: `CRIBL_DIST_MASTER_URL=tls://<authToken>@leader:42(` 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 (ephemeral instances. When set, this environment variable overrides $CRIBL_HOME. It also creates predefined folders in the specified directory 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 Note 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 Worker 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.
`CRIBL_HOST`	The Host URL for authentication. Example: `CRIBL_HOST=<url>` `CRIBL_USERNAME=<username> CRIBL_PASSWORD=<password>`

Name	Purpose
	`$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 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 (Stream, Edge) the Worker 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`.

To generate a random authentication token, leave `<authToken>` unchanged. You can define it to add your own token instead, but make sure it's secure enough.

CRIBL_TMP_DIR

Sources use this variable to construct temporary directories in which to stage downloaded Parquet data. If `CRIBL_TMP_DIR` is not set (the default), Cribl applications create subdirectories within your operating system's default temporary directory:

- For Cribl Stream: `<OS_default_temporary_directory>/stream/`.
- For Cribl Edge: `<OS_default_temporary_directory>/edge/`.

For example, on Linux, Stream's default staging directory would be `/tmp/stream/`.

If you explicitly set this `CRIBL_TMP_DIR` environment variable, its value replaces this OS-specific default parent directory.

CRIBL_DIST_WORKER_PROXY

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 Stream applies this configuration only upon its first startup.

For any method, Cribl Stream 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 Stream 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`
`CRIBL_DIST_MASTER_FAILOVER_VOLUME=/tmp/shared`	Sets the location of the NFS directory to supp failover.
`CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`	Determines how many Lease refresh periods the standby Nodes attempt to promote them primary. Cribl recommends setting this to `3`.
`CRIBL_DIST_MASTER_FAILOVER_PERIOD`	Determines how often the primary Leader ref on the Lease file. Cribl recommends setting th
`CRIBL_INSTANCE_HOME`	In `Failover` mode, this variable points to th `root` directory, as opposed to the shared vol to access `$CRIBL_INSTANCE_HOME/local/_system/` (`C:\Program Data\Cribl\local\_syster` Edge on Windows). Outside of `Failover` mo same value as `CRIBL_CONF_DIR`.

GitOps

Cribl Stream 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 Stream 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`.

Name	Purpose
`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 Worker Groups.
`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`	Specifies the base path where events from various Sources and Destinations are spooled. Defaults to `$CRIBL_HOME/state/spool` or `$CRIBL_VOLUME_DIR/state/spool`.

19.6. Config Files

Understanding Configuration Paths and Files

Even though all Cribl Stream Routes, Pipelines, and Functions can be managed from the UI, it's important to understand how the configuration works under the hood. Here is how configuration paths and files are laid out on the filesystem.

Path Placeholder	Expanded Path
`$CRIBL_HOME`	Standalone Install: `/path/to/install/cribl/` – referred to below as `$CRIBL_HOME` Cribl App for Splunk Install: `$SPLUNK_HOME/etc/apps/cribl/`

All paths below are relative to `$CRIBL_HOME` in a single-instance deployment, or to `$CRIBL_HOME/groups/<group-name>/` in a distributed deployment.

Category	Relative Path
Default Configurations Out-of-the-box defaults (rewritable) and libraries (expandable)	`default/cribl`
Local Configurations User-created integrations and resources	`local/cribl`
System Configuration	`(default|local)/cribl/cribl.yml` See cribl.yml
API Configuration	`(default|local)/cribl/cribl.yml > [api]` section See cribl.yml
Source Configuration	`(default|local)/cribl/inputs.yml` See inputs.yml
Destination Configuration	`(default|local)/cribl/outputs.yml` See outputs.yml
License Configuration	`(default|local)/cribl/licenses.yml`

Category	Relative Path
Regexes Configuration	`(default│local)/cribl/regexes.yml`
Breakers Configuration	`(default│local)/cribl/breakers.yml`
Limits Configuration	`(default│local)/cribl/limits.yml`
Service Processes Configuration	`(default│local)/cribl/service.yml` See service.yml
Pipelines Configuration	`(default│local)/cribl/pipelines/<pipeline_name>` Each Pipeline's config resides within its subdirectory.
Packs Configuration	`default/<pack_name>` Each Pack's code and config reside within its subdirectory.
Routes Configuration	`(default│local)/cribl/pipelines/routes.yml`
Functions	`(default│local)/cribl/functions/<function_name>` Each Function's code resides within its subdirectory.
Functions Configuration	`(default│local)/cribl/functions/<function_name>/...` Each Function's config resides within its subdirectory.
Roles Configuration	`(default│local)/cribl/roles.yml` RBAC Role definitions. See roles.yml.
Policies Configuration	`(default│local)/cribl/policies.yml` RBAC Policy definitions. See policies.yml.
Permissions Configuration	`(default│local)/cribl/perms.yml` User permissions. See perms.yml.
Secrets Configuration	`(default│local)/cribl/secrets.yml` Cribl Stream secrets. See secrets.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, Worker 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 Worker Node.

Upon restarts, be aware of the following:

- Syslog data still being received over UDP might be dropped.
- Worker Nodes will temporarily disappear from the Leader's **Manage Workers** or **Manage Edge Nodes** page.
- Aggregation and suppression operations will start over.
- Worker Nodes' local copies of Monitoring metrics will be dropped.
- Cribl Stream will drop any events still in RAM that were bound for persistent queues. (However, PQ data already written to disk will persist through the restart.)

Changes that require reloads include configuration changes to:

- Functions
- Pipelines
- Packs
- Routes
- Lookups
- Parquet schemas
- Global variables
- Group Settings > Limits
- Group Settings > Logging > Levels

Changes that require restarts include configuration changes to:

- Distributed mode (Leader versus Managed Worker Node or Single instance)
- Worker Group assignment
- Event Breakers
- QuickConnect configs
- Sources
- Destinations
- Group Settings > General Settings > TLS
- Group Settings > General Settings > Advanced
- Group Settings > Worker Processes > Process count and Memory

Some general guidelines to keep in mind:

- Configuration changes generated by most UI interactions – for instance, changing the order of Functions in a Pipeline, or changing the order of Routes – **do not require restarts**.

- Some configuration changes in the **Settings** UI **do require restarts**. These will prompt you for confirmation before restarting.
- All direct edits of configuration files in `(bin|local|default)/cribl/...` **will require restarts**.
- Worker Nodes might temporarily disappear from the Leader's **Workers** or **Edge Nodes** tab while restarting.
- A `git commit` command on the Leader Node's host (using a freestanding `git` client not embedded in Cribl's CLI or UI) **will require either a reload or restart**.
- When using the Cribl App for Splunk, changes to Splunk configuration files might or might not require restarts. Please check current Splunk docs.

Configuration Layering and Precedence

As on most *nix systems, Cribl configurations in `local` take precedence over those in `default`. There is no layering of configuration files.

Editing Configuration Files Manually

When config files **must** be edited manually, save all changes in `local`.

19.6.1. Cribl.Yml

`cribl.yml` contains settings for configuring API and other system properties.

```
$CRIBL_HOME/default/cribl/cribl.yml
```

```
api: # [object]
  host: # [string] Address to bind to. Defaults to 0.0.0.0.
  port: # [number] Port to listen to. Defaults to 9000.
  retryCount: # [number] Number of times to retry binding to API port. Defaults to
  retrySleepSecs: # [number] Period, in seconds, between consecutive port binding r
  baseUrl: # [string] URL base path from which to serve all assets (useful when beh
  disabled: # [boolean] Flag to enable/disable local UI access. Defaults to false.
  workerRemoteAccess: # [boolean] Enable teleporting to Worker Nodes'. Defaults to
  revokeOnRoleChange: # [boolean] Log users out when their roles change. Defaults t
  authTokenTTL: # [number] How long (in seconds) authentication tokens remain valid
  idleSessionTTL: # [number] How long (in seconds) Cribl Stream will observe no use
  headers: # [object] Custom HTTP headers to be sent with every response.
  loginRateLimit: # [string] Rate limit, expressed as maximum number of requests pe
  ssoRateLimit: # [string] Rate limit for SSO and SLO callback endpoints. Expressed
  apiCache: # [object] Toggle to 'No' to disable caching of browser's frequent API
  ssl: # [object] SSL Settings
    disabled: # [boolean] SSL is enabled by default.
    privKeyPath: # [string] Path to private key.
    certPath: # [string] Path to certificate.

auth: # [object] Authentication Settings
  type: # [string] Type - Select one of the supported authentication providers.

  # -------------- if type is ldap ---------------

  secure: # [boolean] Secure - Enable to use a secure ldap connection (ldaps://), c
  ldapServers: # [array of strings] LDAP servers - List of LDAP servers, each entry
  bindDN: # [string] Bind DN - Distinguished name of entity to authenticate with LD
  bindCredentials: # [string] Password - Distinguished Name password used to authen
  searchBase: # [string] User search base - Starting point to search LDAP for users
  usernameField: # [string] Username field - LDAP user search field, e.g. cn or uid
  searchFilter: # [string] User search filter - LDAP search filter to apply when fi
  groupSearchBase: # [string] Group search base - Starting point to search LDAP for
  groupMemberField: # [string] Group member field - LDAP group search field, e.g. m
  groupSearchFilter: # [string] Group search filter - LDAP search filter to apply w
  groupField: # [string] Group name field - LDAP group field, e.g. cn
  connectTimeout: # [number] Connection timeout (ms)
  rejectUnauthorized: # [boolean] Reject unauthorized - Valid for secure LDAP conne
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if L
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not explicit
    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 inst
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 auth
port: # [number] Port - Port of Splunk instance that provides authentication. Def
ssl: # [boolean] SSL - Whether SSL is enabled on Splunk instance that provides au
fallback: # [boolean] Fallback on fatal error - Attempt local authentication if S
groups: # [object]
  default: # [string] Default role - Default role assigned to groups not explicit
  mapping: # [object] Mapping - Group(s)-to-role(s) mappings

# ---------------------------------------------------------



# -------------- if type is openid ---------------

name: # [string] Provider name - The name of the identity provider service. Manua
audience: # [string] Audience - The Audience from provider configuration. This wi
client_id: # [string] Client ID - The client_id from provider configuration.
client_secret: # [string] Client secret - The client_secret provider configuratio
scope: # [string] Scope - Space-separated list of authentication scopes. Default
auth_url: # [string] Authentication URL - The full path to the provider's authent
token_url: # [string] Token URL - The full path to the provider's access token UR
userinfo_url: # [string] User Info URL - The full path to the provider's user inf
logout_url: # [string] Logout URL - The full path to the provider's logout URL. I
userIdExpr: # [string] User identifier - Expression used to derive userId from th
rejectUnauthorized: # [boolean] Validate certs - Validate certificates; set to fa
filter_type: # [string] Filter type - Optional method for limiting access per use
groupField: # [string] Group name field - Field on the id_token that contains the
fallback: # [boolean] Allow local auth - Allows locally configured users to log i
groups: # [object]
```

```
    default: # [string] Default role - Default role assigned to groups not explicit
    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 TLS
  maxVersion: # [string] Maximum TLS version - Maximum TLS version. Defaults to TLS
  defaultCipherList: # [string] Default cipher list - Default suite of enabled and
      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 li
  rejectUnauthorized: # [boolean] Validate server certs - Whether to validate serve
proxy: # [object]
  useEnvVars: # [boolean]
git: # [object]
  branch: # [string] Branch - The branch to track in your Stream deployment's git r
```

```
    gitOps: # [string] GitOps workflow - The GitOps workflow for managing Stream's co
    commitDeploySingleAction: # [boolean] Collapse actions - When enabled, Commit & D
    defaultCommitMessage: # [string] Default commit message - Enter a default message
    remote: # [string] Remote URL - Enter remote git repo's URL.
    authType: # [string] Authentication type - Git authentication type.

    # -------------- if authType is ssh ---------------

    sshKey: # [string] SSH private key - Enter SSH private key (without passphrase) f
    strictHostKeyChecking: # [boolean] SSH strict host key checking - Validate key ag

    # --------------------------------------------------------



    # -------------- if authType is basic ---------------

    user: # [string] User - Username for authentication.
    password: # [string] Password - Password for authentication. (With GitHub, use a

    # --------------------------------------------------------

    autoAction: # [string] Scheduled global actions - Global git actions to run autom

    # -------------- if autoAction is commit ---------------

    autoActionSchedule: # [string] Schedule - Cron schedule to run selected git actio
    autoActionMessage: # [string] Commit message - Default scheduled commit message.

    # --------------------------------------------------------



    # -------------- if autoAction is push ---------------

    autoActionSchedule: # [string] Schedule - Cron schedule to run selected git actio

    # --------------------------------------------------------

    timeout: # [number] Git timeout - Max time (milliseconds) to wait for git process
```

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
```

19.6.2. 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
      type: # [string] Event Breaker Type - Event Breaker Type
      timestampAnchorRegex: # [string] Timestamp Anchor - Regex to match before at
      timestamp: # [object] Timestamp Format - Auto, manual format (strptime) or cu
      type: # [string] Timestamp Type
      length: # [number] Length
      format: # [string] Format
      timestampTimezone: # [string] Default timezone - Timezone to assign to timest
      timestampEarliest: # [string] Earliest timestamp allowed - The earliest times
      timestampLatest: # [string] Future timestamp allowed - The latest timestamp
      maxEventBytes: # [number] Max Event Bytes - The maximum number of bytes that
      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 fiel
      disabled: # [boolean] Disabled - Allows breaker rule to be enabled or disable
```

19.6.3. 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. Opt
  cert: # [string] Certificate - Drag/drop or upload host certificate, in PEM/Base6
  privKey: # [string] Private key - Certificate private key.
  passphrase: # [string] Passphrase - Passphrase. Optional.
  ca: # [string] CA certificate - Optionally, drag/drop or upload all CA certifica
  inUse: # [array of strings] Referenced - List of configurations referencing this
```

19.6.4. 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 t
  onPrem: # [boolean] On prem - Whether the group accepts on-prem or cloud worker n
  isFleet: # [boolean] Is Fleet - Fleet groups only manage edge nodes.
  workerRemoteAccess: # [boolean] UI access - Enable authenticated viewing of Worke
```

19.6.5. 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
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of t
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ----------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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) ar
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thro
    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, e
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
    environment: # [string] Environment - Optionally, enable this config only on a
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ----------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send c
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
kafka_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Kafka brokers to use, eg. local
  topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning
  groupId: # [string] Group ID - Specifies the consumer group this instance belon
  fromBeginning: # [boolean] From beginning - Whether to start reading from earl
  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 Co
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that a
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
      certificateName: # [string] Certificate name - The name of the predefined c
      caPath: # [string] CA certificate path - Path on client in which to find CA
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
      certPath: # [string] Certificate path (mutual auth) - Path on client in wh
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private ke
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use whe
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use whe

      # --------------------------------------------------------



    # --------------------------------------------------------

  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
  sasl: # [object] Authentication - Authentication parameters to use when connect
    disabled: # [boolean] Disabled - Enable Authentication

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

    # --------------------------------------------------------
```

```
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
  certificateName: # [string] Certificate name - The name of the predefined cer
  caPath: # [string] CA certificate path - Path on client in which to find CA c
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
  certPath: # [string] Certificate path (mutual auth) - Path on client in which
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

  # --------------------------------------------------------

sessionTimeout: # [number] Session timeout (ms) -
  Timeout used to detect client failures when using Kafka's group management fa
  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 rebala
  Value must be between the broker's configured group.min.session.timeout.ms an
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_se
rebalanceTimeout: # [number] Rebalance timeout (ms) -
  Maximum allowed time for each worker to join the group after a rebalance has
  If the timeout is exceeded, the coordinator broker will remove the worker fro
  See details [here](https://kafka.apache.org/documentation/#connectconfigs_reb
heartbeatInterval: # [number] Heartbeat interval (ms) -
  Expected time between heartbeats to the consumer coordinator when using Kafka
  Value must be lower than sessionTimeout, and typically should not exceed 1/3
  See details [here](https://kafka.apache.org/documentation/#consumerconfigs_he
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, e
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
  port: # [number] [required] Port - Port to listen to.
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # ----------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to ad
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  criblAPI: # [string] Cribl HTTP Event API - Absolute path on which to listen f
  elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path o
  splunkHecAPI: # [string] Splunk HEC Endpoint - Absolute path on which listen f
  splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC ackn
  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, 
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to 
    commitFrequency: # [number] Commit frequency - The number of events to send 
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # --------------------------------------------------------

  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
  maxActiveCxn: # [number] Max Active Connections - Maximum number of active conn
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
  authTokens: # [array] Auth tokens - Shared secrets to be provided by any Splunk
    - token: # [string] Token - Shared secrets to be provided by any Splunk forwa
      description: # [string] Description - Optional token description
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
splunk_search_input: # [object]
  searchHead: # [string] Search head - Search head base URL, can be expression, d
  search: # [string] [required] Search - Enter Splunk search here. For example:
  earliest: # [string] Earliest - The earliest time boundary for the search. Can
  latest: # [string] Latest - The latest time boundary for the search. Can be an
  cronSchedule: # [string] [required] Cron schedule - A cron schedule on which to
  endpoint: # [string] [required] Search endpoint - REST API used to create a sea
  outputMode: # [string] [required] Output mode - Format of the returned output
  endpointParams: # [array] Endpoint parameters - Optional request parameters to
    - name: # [string] Name - Parameter name
      value: # [string] Value - JavaScript expression to compute the parameter's
  endpointHeaders: # [array] Endpoint headers - Optional request headers to send
    - name: # [string] Name - Header Name
      value: # [string] Value - JavaScript expression to compute the header's val
  logLevel: # [string] Log level - Collector runtime log Level (verbosity).
  requestTimeout: # [number] Request timeout (seconds) - HTTP request inactivity
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS l
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep A
  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, e
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
  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

# --------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select (or create) a stored text s

# --------------------------------------------------------

type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
```

```
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send c
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 addres
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array] Auth tokens - Shared secrets to be provided by any client
      - token: # [string] Token - Shared secret to be provided by any client (Autho
        description: # [string] Description - Optional token description
        metadata: # [array] Fields - Fields to add to events referencing this token
        - name: # [string] Name - Field name
          value: # [string] Value - JavaScript expression to compute field's value,
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined cer
      privKeyPath: # [string] Private key path - Path on server containing the priv
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificate
      caPath: # [string] CA certificate path - Path on server containing CA certifi
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active request
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
    activityLogSampleRate: # [number] Activity log sample rate - How often request
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
    splunkHecAPI: # [string] [required] Splunk HEC Endpoint - Absolute path on whic
    metadata: # [array] Fields - Fields to add to every event. May be overridden by
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, e
    allowedIndexes: # [array of strings] Allowed Indexes - List values allowed in H
    splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC ackno
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send c
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  azure_blob_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The storage account queue name blob notifications
    fileFilter: # [string] Filename filter - Regex matching file names to download
    visibilityTimeout: # [number] Visibility timeout (secs) - The duration (in seco
    numReceivers: # [number] Num receivers - The Number of receiver processes to ru
    maxMessages: # [number] Max messages - The maximum number of messages to return
```

```
servicePeriodSecs: # [number] Service period (secs) - The duration (in seconds
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
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, e
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size
parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds
authType: # [string] Authentication method - Enter connection string directly,
connectionString: # [string] Connection string - Enter your Azure Storage accou

# -------------- if authType is manual ---------------



# --------------------------------------------------------

textSecret: # [string] Connection string (text secret) - Select (or create) a s

# -------------- if authType is secret ---------------



# --------------------------------------------------------

disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
```

```
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # ----------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  elasticAPI: # [string] [required] Elasticsearch API endpoint - Absolute path on
  authType: # [string] Authentication type - Elastic authentication type

  # -------------- if authType is basic ---------------

  username: # [string] Username - Username for Basic authentication
  password: # [string] Password - Password for Basic authentication
```

```
# ---------------------------------------------------------


# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select (or create) a secret

# ---------------------------------------------------------


# -------------- if authType is authTokens ---------------

authTokens: # [array of strings] Token - Bearer tokens to include in the autho

# ---------------------------------------------------------

apiVersion: # [string] API Version - The API version to use for communicating w

# -------------- if apiVersion is custom ---------------

customAPIVersion: # [string] Custom API Version - Custom version information to

# ---------------------------------------------------------

extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
  - name: # [string] Name - Field name
    value: # [string] Value - Field value
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, e
proxyMode: # [object]
  enabled: # [boolean] Enable Proxy Mode - Enable proxying of non-bulk API requ

  # -------------- if enabled is true ---------------

  url: # [string] Proxy URL - URL of the Elastic server to proxy non-bulk reque
  removeHeaders: # [array of strings] Remove headers - List of headers to remov
  timeoutSec: # [number] Proxy request timeout - Amount of time, in seconds, to
  authType: # [string] Authentication method - Enter credentials directly, or s

  # ---------------------------------------------------------
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to 
    commitFrequency: # [number] Commit frequency - The number of events to send 
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
confluent_cloud_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to use
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that ar
    servername: # [string] Server name (SNI) - Server name for the SNI (Server N
    certificateName: # [string] Certificate name - The name of the predefined ce
    caPath: # [string] CA certificate path - Path on client in which to find CA 
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in w
    certPath: # [string] Certificate path (mutual auth) - Path on client in whic
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # ----------------------------------------------------------

  topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning:
  groupId: # [string] Group ID - Specifies the consumer group this instance belon
  fromBeginning: # [boolean] From beginning - Whether to start reading from earli
  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 Con
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that a
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
      certificateName: # [string] Certificate name - The name of the predefined c
      caPath: # [string] CA certificate path - Path on client in which to find CA
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
      certPath: # [string] Certificate path (mutual auth) - Path on client in whi
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private ke
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use whe
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use whe

      # ----------------------------------------------------------


    # ----------------------------------------------------------

  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
  sasl: # [object] Authentication - Authentication parameters to use when connect
    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 fa
    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 rebala
    Value must be between the broker's configured group.min.session.timeout.ms an
    See details [here](https://kafka.apache.org/documentation/#consumerconfigs_se
  rebalanceTimeout: # [number] Rebalance timeout (ms) -
    Maximum allowed time for each worker to join the group after a rebalance has
    If the timeout is exceeded, the coordinator broker will remove the worker fro
    See details [here](https://kafka.apache.org/documentation/#connectconfigs_reb
  heartbeatInterval: # [number] Heartbeat interval (ms) -
    Expected time between heartbeats to the consumer coordinator when using Kafka
    Value must be lower than sessionTimeout, and typically should not exceed 1/3
    See details [here](https://kafka.apache.org/documentation/#consumerconfigs_he
  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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
```

```
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
  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 ce
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # ----------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  prometheusAPI: # [string] Remote Write API endpoint - Absolute path on which to
  lokiAPI: # [string] Logs API endpoint - Absolute path on which to listen for Lo
  keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wai
  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 heade

    # -------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secre

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text

    # -------------------------------------------------------

  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 heade

    # -------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secre
    credentialsSecret: # [string] Credentials secret - Select (or create) a secre

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text

    # -------------------------------------------------------

  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, e
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send o
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
loki_input: # [object]
  type: # [string] Input Type
```

```
type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addres
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 ce
  privKeyPath: # [string] Private key path - Path on server containing the pri
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificat
  caPath: # [string] CA certificate path - Path on server containing CA certif
  requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requ
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept f
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept f

  # --------------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active reques
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to ad
activityLogSampleRate: # [number] Activity log sample rate - How often request
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
lokiAPI: # [string] [required] Logs API endpoint - Absolute path on which to l
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
# -------------- if authType is credentialsSecret ---------------

credentialsSecret: # [string] Credentials secret - Select (or create) a secret

# -------------------------------------------------------------



# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select (or create) a stored text

# -------------------------------------------------------------


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,
pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to
  commitFrequency: # [number] Commit frequency - The number of events to send
  maxFileSize: # [string] Max file size - The maximum size to store in each qu
  maxSize: # [string] Max queue size - The maximum amount of disk space the qu
  path: # [string] Queue file path - The location for the persistent queue fil
  compress: # [string] Compression - Codec to use to compress the persisted da

# -------------------------------------------------------------

streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # --------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  prometheusAPI: # [string] [required] Remote Write API endpoint - Absolute path
  keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wai
  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

# -------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select (or create) a stored text s

# -------------------------------------------------------

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, e
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
  commitFrequency: # [number] Commit frequency - The number of events to send d
  maxFileSize: # [string] Max file size - The maximum size to store in each que
  maxSize: # [string] Max queue size - The maximum amount of disk space the que
  path: # [string] Queue file path - The location for the persistent queue file
  compress: # [string] Compression - Codec to use to compress the persisted dat
  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
prometheus_input: # [object]
  dimensionList: # [array of strings] Extra Dimensions - Other dimensions to inc
  discoveryType: # [string] Discovery Type - Target discovery mechanism. Use sta

  # -------------- if discoveryType is static ---------------

  targetList: # [array of strings] Targets - List of Prometheus targets to pull

  # ---------------------------------------------------------



  # -------------- 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
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from

  # ---------------------------------------------------------



  # -------------- if discoveryType is ec2 ---------------

  usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting
  scrapePort: # [number] Metrics Port - The port number in the metrics URL for d
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
  searchFilter: # [array] Search Filter - EC2 Instance Search Filter
    - Name: # [string] Filter Name - Search filter attribute name, see: https://
      Values: # [array of strings] Filter Values - Search Filter Values, if empt
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication
  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 AW
  signatureVersion: # [string] Signature version - Signature version to use for
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
  enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the r
```

```
assumeRoleExternalId: # [string] External ID - External ID to use when assuming

# ----------------------------------------------------------


interval: # [number] Poll Interval - How often in minutes to scrape targets for
logLevel: # [string] [required] Log Level - Collector runtime Log Level
keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep A
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, e
authType: # [string] Authentication method - Enter credentials directly, or sel
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

# -------------- 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 befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue
```

```
  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep A
  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, e
  planType: # [string] [required] Subscription Plan - Office 365 subscription pla
  publisherIdentifier: # [string] Publisher Identifier - Optional Publisher Ident
  contentConfig: # [array] Content Types - Enable Office 365 Management Activity
    - contentType: # [string] Content Type - Office 365 Management Activity API C
      description: # [string] Interval Description - If interval type is minutes
      interval: # [number] Interval
      logLevel: # [string] Log Level - Collector runtime Log Level
      enabled: # [boolean] Enabled
  authType: # [string] Authentication method - Enter client secret directly, or s
  clientSecret: # [string] Client secret - Office 365 Azure client secret

  # -------------- if authType is manual ---------------



  # ----------------------------------------------------

  textSecret: # [string] Client secret (text secret) - Select (or create) a store

  # -------------- if authType is secret ---------------
  # ----------------------------------------------------------

  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ----------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to
    commitFrequency: # [number] Commit frequency - The number of events to send
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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
  keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
  maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep
  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,
  contentConfig: # [array] Content Types - Enable Office 365 Service Communicatio
    - contentType: # [string] Content Type - Office 365 Services API Content Type
      description: # [string] Interval Description - If interval type is minutes
      interval: # [number] Interval
      logLevel: # [string] Log Level - Collector runtime Log Level
      enabled: # [boolean] Enabled
  authType: # [string] Authentication method - Enter client secret directly, or s
  clientSecret: # [string] Client secret - Office 365 Azure client secret

  # -------------- if authType is manual ---------------



  # --------------------------------------------------------

  textSecret: # [string] Client secret (text secret) - Select (or create) a store

  # -------------- if authType is secret ---------------



  # --------------------------------------------------------

  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations, 
    - 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 
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
```

```
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 t
  startDate: # [string] Date range start - Backward offset for the search range's
  endDate: # [string] Date range end - Backward offset for the search range's ta
  logLevel: # [string] Log level - Log Level (verbosity) for collection runtime b
  timeout: # [number] Timeout (secs) - HTTP request inactivity timeout. Maximum i
  disableTimeFilter: # [boolean] Disable time filter - Disables time filtering of
  authType: # [string] Authentication method - Select authentication method.

  # -------------- if authType is manual ---------------

  username: # [string] Username - Username to run Message Trace API call.
  password: # [string] Password - Password to run Message Trace API call.

  # ----------------------------------------------------------



  # -------------- if authType is secret ---------------

  credentialsSecret: # [string] Credentials secret - Select (or create) a secret

  # ----------------------------------------------------------



  # -------------- if authType is oauth ---------------

  clientSecret: # [string] Client secret - client_secret to pass in the OAuth req
  tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Mi
  clientId: # [string] Client ID - client_id to pass in the OAuth request paramet
  resource: # [string] Resource - Resource to pass in the OAuth request paramete

  # ----------------------------------------------------------



  # -------------- if authType is oauthSecret ---------------
```

```
textSecret: # [string] Client secret - Select (or create) a secret that referer
tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Mi
clientId: # [string] Client ID - client_id to pass in the OAuth request paramet
resource: # [string] Resource - Resource to pass in the OAuth request parameter

# ----------------------------------------------------------

keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should
maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep A
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, e
type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ----------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ----------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
  commitFrequency: # [number] Commit frequency - The number of events to send c
  maxFileSize: # [string] Max file size - The maximum size to store in each que
  maxSize: # [string] Max queue size - The maximum amount of disk space the que
  path: # [string] Queue file path - The location for the persistent queue file
  compress: # [string] Compression - Codec to use to compress the persisted dat

# ----------------------------------------------------------

streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
eventhub_input: # [object]
  type: # [string] Input Type
  brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to cor
  topics: # [array of strings] [required] Event Hub name - The name of the Event
  groupId: # [string] Group ID - Specifies the consumer group this instance belor
  fromBeginning: # [boolean] From beginning - Whether to start reading from earl
  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
  sasl: # [object] Authentication - Authentication parameters to use when connect
    disabled: # [boolean] Disabled - Enable authentication.

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
    username: # [string] Username - The username for authentication. For Event Hu
    authType: # [string] Authentication method - Enter password directly, or sele

    # ---------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this

    # ---------------------------------------------------------

  sessionTimeout: # [number] Session timeout (ms) -
    Timeout (a.k.a session.timeout.ms in Kafka domain) used to detect client fail
    If the client sends the broker no heartbeats before this timeout expires, the
    Value must be lower than rebalanceTimeout.
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/r
  rebalanceTimeout: # [number] Rebalance timeout (ms) -
    Maximum allowed time (a.k.a rebalance.timeout.ms in Kafka domain) for each wo
    If the timeout is exceeded, the coordinator broker will remove the worker fro
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/r
  heartbeatInterval: # [number] Heartbeat interval (ms) -
    Expected time (a.k.a heartbeat.interval.ms in Kafka domain) between heartbeat
    Value must be lower than sessionTimeout, and typically should not exceed 1/3
    See details [here](https://github.com/Azure/azure-event-hubs-for-kafka/blob/r
  minimizeDuplicates: # [boolean] Minimize duplicates - Enable feature to minimiz
  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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ----------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
exec_input: # [object]
  disabled: # [boolean] Disabled - Enable/disable this input
  command: # [string] Command - Command to execute; supports Bourne shell syntax
  retries: # [number] Max retries - Maximum number of retry attempts in the event
  scheduleType: # [string] Schedule type - Select a schedule type; either an inte

  # -------------- 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
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of t
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
  type: # [string] Input Type
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
port: # [number] [required] Port - Port to listen to.
authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined ce
  privKeyPath: # [string] Private key path - Path on server containing the pri
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  certPath: # [string] Certificate path - Path on server containing certificat
  caPath: # [string] CA certificate path - Path on server containing CA certif
  requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requ
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept f
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept f

  # ---------------------------------------------------------

maxActiveReq: # [number] Max active requests - Maximum number of active reques
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to ad
activityLogSampleRate: # [number] Activity log sample rate - How often request
requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
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, 
pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------
```

```
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 subscriptio
  createTopic: # [boolean] Create topic - If enabled, create topic if it does not
  createSubscription: # [boolean] Create subscription - If enabled, create subscr

  # -------------- if createSubscription is true ---------------

  orderedDelivery: # [boolean] Ordered delivery - If enabled, receive events in t

  # ----------------------------------------------------------

  region: # [string] Region - Region to retrieve messages from. Select 'default'
  googleAuthMethod: # [string] Authentication Method - Google authentication meth

  # -------------- if googleAuthMethod is manual ---------------

  serviceAccountCredentials: # [string] Service account credentials - Contents of

  # ----------------------------------------------------------


  # -------------- if googleAuthMethod is secret ---------------

  secret: # [string] Service account credentials (text secret) - Select (or creat

  # ----------------------------------------------------------

  maxBacklog: # [number] Max backlog - If Destination exerts backpressure, this s
  requestTimeout: # [number] Request timeout (ms) - Pull request timeout, in mil
  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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------
  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ----------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr
    # -------------------------------------------------------

  maxActiveCxn: # [number] Max Active Connections - Maximum number of active conn
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
  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, e
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
  port: # [number] [required] Port - Port to listen to.
  authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    certificateName: # [string] Certificate name - The name of the predefined cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # --------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  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, e
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
```

```
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
  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 ce
    privKeyPath: # [string] Private key path - Path on server containing the pri
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    certPath: # [string] Certificate path - Path on server containing certificat
    caPath: # [string] CA certificate path - Path on server containing CA certif
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requ
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept f
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept f

    # ----------------------------------------------------------

  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses
  maxActiveCxn: # [number] Max Active Connections - Maximum number of active con
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
  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,
  authType: # [string] Authentication method - Enter a token directly, or provid
  authToken: # [string] Auth token - Shared secret to be provided by any client

  # -------------- if authType is manual ---------------



  # ----------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t
  textSecret: # [string] Auth token (text secret) - Select (or create) a stored

  # -------------- if authType is secret ---------------


  # -------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to
    commitFrequency: # [number] Commit frequency - The number of events to send
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
system_metrics_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive
  host: # [object]
    mode: # [string]  - Select level of detail for host metrics

    # -------------- if mode is custom ---------------

    custom: # [object]
      system: # [object]
```

```
system: # [object]
  mode: # [string]  - Select the level of details for system metrics

  # -------------- if mode is custom ---------------

  processes: # [boolean] Process metrics - Generate metrics for the numbers

  # ---------------------------------------------------------

cpu: # [object]
  mode: # [string]  - Select the level of details for CPU metrics

  # -------------- if mode is custom ---------------

  perCpu: # [boolean] Per CPU metrics - Generate metrics for each CPU
  detail: # [boolean] Detailed metrics - Generate metrics for all CPU state
  time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time cou

  # ---------------------------------------------------------

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 st

  # ---------------------------------------------------------

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 in
  perInterface: # [boolean] Per interface metrics - Generate separate metri
  detail: # [boolean] Detailed metrics - Generate full network metrics

  # ---------------------------------------------------------

disk: # [object]
  mode: # [string]  - Select the level of details for disk metrics

  # -------------- if mode is custom ---------------
```

```
    # -------------- if mode is custom ---------------

    devices: # [array of strings] Device filter - Block devices to include/ex
    mountpoints: # [array of strings] Mountpoint filter - Filesystem mountpo
    fstypes: # [array of strings] Filesystem type filter - Filesystem types 
    perDevice: # [boolean] Per device metrics - Generate separate metrics fo
    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 U
  dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Dock

  # --------------------------------------------------------


  # -------------- if mode is all ---------------

  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's U
  dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Dock

  # --------------------------------------------------------


  # -------------- if mode is custom ---------------

  filters: # [array] Container Filters - Containers matching any of these will
    - expr: # [string] Expression
  allContainers: # [boolean] All containers - Include stopped and paused conta
  perDevice: # [boolean] Per device metrics - Generate separate metrics for ea
  detail: # [boolean] Detailed metrics - Generate full container metrics
  dockerSocket: # [array of strings] Docker socket - Full paths for Docker's U
  dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Dock

  # --------------------------------------------------------
```

```
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, e
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 cons
  maxDataTime: # [string] Max data age - Maximum amount of time to retain data
  compress: # [string] Compression - Select data compression format. Optional.
  destPath: # [string] Path location - Path to use to write metrics. Defaults t

  # -------------------------------------------------------

type: # [string] Input Type
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
  commitFrequency: # [number] Commit frequency - The number of events to send d
  maxFileSize: # [string] Max file size - The maximum size to store in each que
  maxSize: # [string] Max queue size - The maximum amount of disk space the que
  path: # [string] Queue file path - The location for the persistent queue file
  compress: # [string] Compression - Codec to use to compress the persisted dat
  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
system_state_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive s
  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, e
  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 cons
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults t

    # -------------------------------------------------------

  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

      # [object]
  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send (
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
windows_metrics_input: # [object]
  interval: # [number] Polling interval - Time, in seconds, between consecutive m
  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 in

        # --------------------------------------------------------

      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 state
        time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time cou

        # --------------------------------------------------------

      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 st

        # ---------------------------------------------------------

      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 in
        perInterface: # [boolean] Per interface metrics - Generate separate metri
        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,
        perVolume: # [boolean] Per volume metrics - Generate separate metrics for

        # ---------------------------------------------------------


    # ---------------------------------------------------------

  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
  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 cons
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults t
```

```
    # ---------------------------------------------------------

  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to 
    commitFrequency: # [number] Commit frequency - The number of events to send 
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
crowdstrike_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read n
  fileFilter: # [string] Filename filter - Regex matching file names to download
  awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Le
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication

  # -------------- 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

# ----------------------------------------------------------

awsSecretKey: # [string] Secret key - Secret key
region: # [string] Region - AWS Region where the S3 bucket and SQS queue are lo
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS
signatureVersion: # [string] Signature version - Signature version to use for s
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
maxMessages: # [number] Max Messages - The maximum number of messages SQS shoul
visibilityTimeout: # [number] Visibility timeout seconds - The duration (in sec
numReceivers: # [number] Num receivers - The Number of receiver processes to ru
socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in second
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to ac
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
assumeRoleExternalId: # [string] External ID - External ID to use when assuming
enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials w
preprocess: # [object]
  disabled: # [boolean] Disabled - Enable Custom Command
  command: # [string] Command - Command to feed the data through (via stdin) an
  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, e
pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for ev
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send (
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 address
    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 cer
      privKeyPath: # [string] Private key path - Path on server containing the priv
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificate
      caPath: # [string] CA certificate path - Path on server containing CA certifi
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active request
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connectio
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to ad
    activityLogSampleRate: # [number] Activity log sample rate - How often request
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
    extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each in
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, 
    proxyMode: # [object]
      enabled: # [boolean] Forward API key validation requests - Toggle to Yes to 
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to 
      commitFrequency: # [number] Commit frequency - The number of events to send 
      maxFileSize: # [string] Max file size - The maximum size to store in each qu
      maxSize: # [string] Max queue size - The maximum amount of disk space the qu
      path: # [string] Queue file path - The location for the persistent queue fil
      compress: # [string] Compression - Codec to use to compress the persisted da

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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. o
    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, e
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send c
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 address
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------
      certificateName: # [string] Certificate name - The name of the predefined cer
      privKeyPath: # [string] Private key path - Path on server containing the priv
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificate
      caPath: # [string] CA certificate path - Path on server containing CA certifi
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active request
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
    activityLogSampleRate: # [number] Activity log sample rate - How often request
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
    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, e
    allowedPaths: # [array of strings] Allowed URI paths - List of URI paths accept
    allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP method
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send d
```

```
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 (
    shardExpr: # [string] Shard selection expression - A JS expression to be callec
    shardIteratorType: # [string] Shard iterator start - Location at which to start
    payloadFormat: # [string] Record data format - Format of data inside the Kinesi
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication

    # -------------- 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

    # ----------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is locat
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defa
    signatureVersion: # [string] Signature version - Signature version to use for s
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role crede
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
    verifyKPLCheckSums: # [boolean] Verify KPL checksums - Verify Kinesis Producer
    avoidDuplicates: # [boolean] Avoid duplicate records - Yes means: when resuming
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send d
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 address
    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 cer
      privKeyPath: # [string] Private key path - Path on server containing the priv
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificate
      caPath: # [string] CA certificate path - Path on server containing CA certifi
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

      # --------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active conn
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, e
    authType: # [string] Authentication method - Enter a token directly, or provide
    authToken: # [string] Auth token - Shared secret to be provided by any client (

    # -------------- if authType is manual ---------------


    # --------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue
```

```
  # -------------- if pqEnabled is true ----------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send (
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
criblmetrics_input: # [object]
  type: # [string] Input Type
  prefix: # [string] Metric Name Prefix - A prefix that is applied to the metrics
  fullFidelity: # [boolean] Full Fidelity - Include granular metrics.  Disabling
  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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ----------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send (
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 th
  udpPort: # [number] UDP Port - Enter UDP port number to listen on. Not required
  tcpPort: # [number] TCP Port - Enter TCP port number to listen on. Not required
  maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  tls: # [object] TLS settings (server side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ----------------

    certificateName: # [string] Certificate name - The name of the predefined cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # ----------------------------------------------------------

  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, e
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
 pqEnabled: # [boolean] Enable Persistent Queue

 # -------------- if pqEnabled is true ---------------

 pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
  commitFrequency: # [number] Commit frequency - The number of events to send d
  maxFileSize: # [string] Max file size - The maximum size to store in each que
  maxSize: # [string] Max queue size - The maximum amount of disk space the que
  path: # [string] Queue file path - The location for the persistent queue file
  compress: # [string] Compression - Codec to use to compress the persisted dat

 # ---------------------------------------------------------

 streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
s3_input: # [object]
 type: # [string] Input Type
 queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read no
 fileFilter: # [string] Filename filter - Regex matching file names to download
 awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Lea
 awsAuthenticationMethod: # [string] Authentication method - AWS authentication

 # -------------- 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

 # ---------------------------------------------------------

 awsSecretKey: # [string] Secret key - Secret key
 region: # [string] Region - AWS Region where the S3 bucket and SQS queue are lo
 endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS
```

```
signatureVersion: # [string] Signature version - Signature version to use for s
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
maxMessages: # [number] Max Messages - The maximum number of messages SQS shoul
visibilityTimeout: # [number] Visibility timeout seconds - The duration (in sec
numReceivers: # [number] Num receivers - The Number of receiver processes to ru
socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in second
skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to ac
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
assumeRoleExternalId: # [string] External ID - External ID to use when assuming
enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials w
preprocess: # [object]
  disabled: # [boolean] Disabled - Enable Custom Command
  command: # [string] Command - Command to feed the data through (via stdin) ar
  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, e
parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size
parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds
pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for ev
disabled: # [boolean] Disabled - Enable/disable this input
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
```

```
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 th
  port: # [number] [required] UDP Port - UDP port to receive SNMP traps on. Defau
  maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events t
  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ----------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 addres
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi

    # -------------------------------------------------------

  maxActiveReq: # [number] Max active requests - Maximum number of active request
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
  activityLogSampleRate: # [number] Activity log sample rate - How often request
  requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
  extractSpans: # [boolean] Extract spans - Toggle to Yes to extract each incomin
  extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each inc
  maxActiveCxn: # [number] Max active connections - Maximum number of active conn
  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

# ----------------------------------------------------------


# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select (or create) a stored text

# ----------------------------------------------------------

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, e
pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
  commitFrequency: # [number] Commit frequency - The number of events to send 
  maxFileSize: # [string] Max file size - The maximum size to store in each que
  maxSize: # [string] Max queue size - The maximum amount of disk space the qu
```

```
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  sqs_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read ev
    queueType: # [string] [required] Queue Type - The queue type used (or created).

    # -------------- if queueType is standard ---------------

    numReceivers: # [number] Num receivers - The Number of receiver processes to ru

    # ---------------------------------------------------------

    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Lea
    createQueue: # [boolean] Create Queue - Create queue if it does not exist.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication

    # -------------- 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

    # ---------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the SQS queue is located. Required
    endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS
    signatureVersion: # [string] Signature version - Signature version to use for s
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
    enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to a
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
    assumeRoleExternalId: # [string] External ID - External ID to use when assumin
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
    maxMessages: # [number] Max Messages - The maximum number of messages SQS shoul
    visibilityTimeout: # [number] Visibility Timeout Seconds - The duration (in sec
    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, e
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for ev
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
      - 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
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
      commitFrequency: # [number] Commit frequency - The number of events to send c
      maxFileSize: # [string] Max file size - The maximum size to store in each que
      maxSize: # [string] Max queue size - The maximum amount of disk space the que
      path: # [string] Queue file path - The location for the persistent queue file
      compress: # [string] Compression - Codec to use to compress the persisted dat

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 th
    udpPort: # [number] UDP port - Enter UDP port number to listen on. Not required
    tcpPort: # [number] TCP port - Enter TCP port number to listen on. Not required
    maxBufferSize: # [number] Max buffer size (events) - Maximum number of events t
    ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses t
```

```
ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses t
timestampTimezone: # [string] Default timezone - Timezone to assign to timestam
singleMsgUdpPackets: # [boolean] Single msg per UDP - Whether to treat UDP pack
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
keepFieldsList: # [array of strings] Fields to keep - Wildcard list of fields t
octetCounting: # [boolean] Octet count framing - Enable if incoming messages us
tls: # [object] TLS settings (server side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  certificateName: # [string] Certificate name - The name of the predefined cer
  privKeyPath: # [string] Private key path - Path on server containing the priv
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  certPath: # [string] Certificate path - Path on server containing certificate
  caPath: # [string] CA certificate path - Path on server containing CA certifi
  requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

  # --------------------------------------------------------

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, e
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
pqEnabled: # [boolean] Enable Persistent Queue

# -------------- if pqEnabled is true ---------------

pq: # [object]
  mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
  maxBufferSize: # [number] Max buffer size - The maximum amount of events to
```

```
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to r
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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
  depth: # [number] Max depth - Set how many subdirectories deep to search. Use 0

  # ---------------------------------------------------------

  interval: # [number] Polling interval - Time, in seconds, between scanning for
  filenames: # [array of strings] Filename allowlist - The full path of discovere
  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, e
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue
  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
  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 cer
    privKeyPath: # [string] Private key path - Path on server containing the priv
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    certPath: # [string] Certificate path - Path on server containing certificate
    caPath: # [string] CA certificate path - Path on server containing CA certifi
    requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
    minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

    # --------------------------------------------------------

  ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
  maxActiveCxn: # [number] Max Active Connections - Maximum number of active conn
  enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
  metadata: # [array] Fields - Fields to add to events from this input.
    - name: # [string] Name - Field name
      value: # [string] Value - JavaScript expression to compute field's value, e
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
  enableHeader: # [boolean] Enable Header - If enabled, client will pass the hea

  # -------------- if enableHeader is true ---------------

  authType: # [string] Authentication method - Enter a token directly, or provid

  # --------------------------------------------------------

  preprocess: # [object]
    disabled: # [boolean] Disabled - Enable Custom Command
    command: # [string] Command - Command to feed the data through (via stdin) a
    args: # [array of strings] Arguments - Arguments
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befo
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directl

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesys
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to 
    commitFrequency: # [number] Commit frequency - The number of events to send 
    maxFileSize: # [string] Max file size - The maximum size to store in each qu
    maxSize: # [string] Max queue size - The maximum amount of disk space the qu
    path: # [string] Queue file path - The location for the persistent queue fil
    compress: # [string] Compression - Codec to use to compress the persisted da

  # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
appscope_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
```

```
ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses t
maxActiveCxn: # [number] Max Active Connections - Maximum number of active conn
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
metadata: # [array] Fields - Fields to add to events from this input.
  - name: # [string] Name - Field name
    value: # [string] Value - JavaScript expression to compute field's value, e
breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
enableUnixPath: # [boolean] UNIX domain socket - Toggle to Yes to specify a fil

# -------------- if enableUnixPath is false ---------------

host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all address
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 cer
  privKeyPath: # [string] Private key path - Path on server containing the priv
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  certPath: # [string] Certificate path - Path on server containing certificate
  caPath: # [string] CA certificate path - Path on server containing CA certifi
  requestCert: # [boolean] Authenticate client (mutual auth) - Whether to requi
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr

  # --------------------------------------------------------



# --------------------------------------------------------



# -------------- if enableUnixPath is true ---------------

unixSocketPath: # [string] UNIX socket path - Path to the UNIX domain socket to
unixSocketPerms: # [string,number] UNIX socket permissions - Permissions to set

# --------------------------------------------------------

authType: # [string] Authentication method - Enter a token directly, or provide
authToken: # [string] Auth token - Shared secret to be provided by any client (
  # -------------- if authType is manual ---------------


  # -------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

  # -------------- if authType is secret ---------------


  # -------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send c
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 address
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 cert
  requestCert: # [boolean] Authenticate client - Required for WEF certificate a
  certificateName: # [string] Certificate name - Name of the predefined certifi
  privKeyPath: # [string] Private key path - Path on server containing the priv
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  certPath: # [string] [required] Certificate path - Path on server containing
  caPath: # [string] [required] CA certificate path - Path on server containing
  commonNameRegex: # [string] Common name - Regex matching allowable common nar
  minVersion: # [string] Minimum TLS version - Minimum TLS version to accept fr
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept fr
maxActiveReq: # [number] Max active requests - Maximum number of active request
enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection
captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
caFingerprint: # [string] CA fingerprint override - SHA1 fingerprint expected b
allowMachineIdMismatch: # [boolean] Allow MachineID mismatch - Allow events to
subscriptions: # [array] [required] Subscriptions - Subscriptions to events on
  - subscriptionName: # [string] Name - Friendly name for this subscription.
    version: # [string] Version - Version UUID for this subscription. If any su
    contentFormat: # [string] Format - Content format in which the endpoint sho
    heartbeatInterval: # [number] Heartbeat - Max time (in seconds) between end
    batchTimeout: # [number] Batch timeout - Interval (in seconds) over which t
    readExistingEvents: # [boolean] Read existing events - Set to Yes if a newl
    sendBookmarks: # [boolean] Use bookmarks - If toggled to Yes, Cribl Stream
    compress: # [boolean] Compression - If toggled to Yes, Stream will receive
    targets: # [array of strings] Targets - Enter the DNS names of the endpoint
    querySelector: # [string] Query builder mode - Select the query builder mod
pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

# -------------- if sendToRoutes is false ---------------

connections: # [array] Quick Connections - Direct connections to Destinations,
  - 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
```

```
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send b
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
win_event_logs_input: # [object]
  type: # [string] Input Type
  logNames: # [array of strings] Event Logs - Enter the event logs to collect. Ru
  readMode: # [string] Read Mode - Read all stored and future event logs, or only
  interval: # [number] Polling Interval - Time, in seconds, between checking for
  batchSize: # [number] Batch Size - Maximum number of event records to read in o
  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, e
  disabled: # [boolean] Disabled - Enable/disable this input
  pipeline: # [string] Pipeline - Pipeline to process data from this Source befor
  sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly

  # -------------- if sendToRoutes is false ---------------

  connections: # [array] Quick Connections - Direct connections to Destinations,
    - 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
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
    mode: # [string] Mode - With Smart mode, PQ will write events to the filesyst
    maxBufferSize: # [number] Max buffer size - The maximum amount of events to h
    commitFrequency: # [number] Commit frequency - The number of events to send d
    maxFileSize: # [string] Max file size - The maximum size to store in each que
    maxSize: # [string] Max queue size - The maximum amount of disk space the que
    path: # [string] Queue file path - The location for the persistent queue file
    compress: # [string] Compression - Codec to use to compress the persisted dat

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

19.6.6. Instance.Yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml` (`C:\Program Data\Cribl\local\_system.yml` for Cribl Edge on Windows).

$CRIBL_HOME/local/_system/instance.yml or C:\Program Data\Cribl\local_system.yml for Cribl Edge on Windows

```
distributed:
  mode: # [string] One of: master | worker | single | edge | managed-edge.
  master: # [object]
    host: # [string] Instance host address.
    port: # [number] Instance post number.
    tls: # [object]
      disabled: # [boolean]
      certificateName: # [string] TLS certificate name.
      rejectUnauthorized: # [boolean] False if ignoring untrusted certificates.
      requestCert: # [boolean] Whether certificates are required and validated.
      privKeyPath: # [string] Path to private key file. Can reference $ENV_VARS.
      certPath: # [string] Path to certificate file. Can reference $ENV_VARS.
      caPath: # [string] Path to CA certificate file. Can reference $ENV_VARS.
      commonNameRegex: # [any] Allowed common names in peer certificates' subject
      # Available values: 'TLSv1.3' | 'TLSv1.2' | 'TLSv1.1' | 'TLSv1'
      minVersion: # [string] Minimum TLS version.
      maxVersion: # [string] Maximum TLS version.
      servername: # [string] Server name.
    resiliency: # [string] Preferred failover mode, one of: none | failover.
    ipWhitelistRegex: # [string] IP addresses allowed to send data (regex).
    maxActiveCxn: # [number] Max number of active connections allowed per Worker P
    authToken: # [string] Token used for communication between Leader and managed
    compression: # [string] Codec used to compress persisted data. One of: none |
    connectionTimeout: # [number] Maximum time to wait for a connection to complet
    writeTimeout: # [number] Time (ms) to wait for a write to complete before assu
    configBundles: # [object]
      remoteUrl: # [string] Bucket to use for remote bundle storage, in s3://${buc
  group: # [string] Group the managed node belongs to.
  envRegex: # [string] Regex used to filter env variable names that can be sent by
  tags: # [array of strings] Tags associated with managed node, can be used by Lea
```

19.6.7. 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 r
  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 Desti
      authType: # [string] Authentication method - Enter authentication data direc

      # -------------- if authType is manual ---------------

      connectionString: # [string] Connection string - Enter your Azure storage ac

      # --------------------------------------------------------



      # -------------- if authType is secret ---------------

      textSecret: # [string] Connection string (text secret) - Text secret

      # --------------------------------------------------------

      containerName: # [string] Container name - Name of the container to collect
      path: # [string] Path - The directory from which to collect data. Templating
      extractors: # [array] Path extractors - Allows using template tokens as cont
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receive
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metada

    # --------------------------------------------------------



    # -------------- if type is filesystem ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - Select a predefined configuratio
      path: # [string] Directory - The directory from which to collect data. Templa
      extractors: # [array] Path extractors - Allows using template tokens as cont
        - key: # [string] Token - A token from the template directory, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (files) - Maximum number of metadata

    # -------------------------------------------------------



    # -------------- if type is google_cloud_storage ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined Destin
      bucket: # [string] Bucket name - Name of the bucket to collect from. This val
      path: # [string] Path - The directory from which to collect data. Templating
      extractors: # [array] Path extractors - Allows using template tokens as conte
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
      endpoint: # [string] Endpoint - Google Cloud Storage service endpoint. If emp
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collecto
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadat
      authType: # [string] Authentication method - Enter account credentials manual

      # -------------- if authType is manual ---------------

      serviceAccountCredentials: # [string] Service account credentials - Contents

      # -------------------------------------------------------



      # -------------- if authType is secret ---------------

      textSecret: # [string] Service account credentials (text secret) - Select (or

      # -------------------------------------------------------



    # -------------------------------------------------------



    # -------------- if type is health_check ---------------

    conf: # [object] [required]
      discovery: # [object]
        discoverType: # [string] Discover Type - Defines how task discovery will be

        # -------------- if discoverType is http ---------------

        discoverUrl: # [string] Discover URL - Expression to derive URL to use for
        discoverMethod: # [string] Discover method - Discover HTTP method.
        discoverRequestHeaders: # [array] Discover Headers - Optional discover requ
          - name: # [string] Name - Header name.
            value: # [string] Value - JavaScript expression to compute the header v
        discoverDataField: # [string] Discover Data Field - Path to field in the re

        # --------------------------------------------------------



        # -------------- if discoverType is json ---------------

        manualDiscoverResult: # [string] Discover result - Allows hard-coding the D
        discoverDataField: # [string] Discover data field - Within the response JSO

        # --------------------------------------------------------



        # -------------- if discoverType is list ---------------

        itemList: # [array of strings] Discover items - Comma-separated list of ite

        # --------------------------------------------------------

      collectUrl: # [string] Health check URL - Expression to derive URL to use for
      collectMethod: # [string] [required] Health check method - Health check HTTP

      # -------------- if collectMethod is get ---------------

      collectRequestParams: # [array] Health check parameters - Optional health che
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter

      # --------------------------------------------------------



      # -------------- if collectMethod is post ---------------

      collectRequestParams: # [array] Health check parameters - Optional health che
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter

      # -------------------------------------------------------


      # -------------- if collectMethod is post_with_body ---------------

      collectBody: # [string] Health check POST Body - Template for POST body to se

      # -------------------------------------------------------

      collectRequestHeaders: # [array] Health check headers - Optional health checl
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header val
      authenticateCollect: # [boolean] Authenticate health check - Enable to make a
      authentication: # [string] [required] Authentication - Authentication method

      # -------------- 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 store

      # -------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST Body - Template for POST body to send with login r
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in l
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to co
      authRequestHeaders: # [array] Authentication Headers - Optional authenticatio
        - name: # [string] Name - Header name.
```

```
    value: # [string] Value - JavaScript expression to compute the header val

# -------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is
credentialsSecret: # [string] Credentials secret - Select (or create) a store
loginBody: # [string] POST Body - Template for POST body to send with login r
tokenRespAttribute: # [string] Token Attribute - Path to token attribute in l
authHeaderExpr: # [string] Authorize Expression - JavaScript expression to co
authRequestHeaders: # [array] Authentication Headers - Optional authenticatio
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header val

# -------------------------------------------------------


# -------------- if authentication is oauth ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
clientSecretParamName: # [string] Client secret parameter - Parameter name th
clientSecretParamValue: # [string] Client secret value - Secret value to add
authRequestParams: # [array] Extra authentication parameters - OAuth request
  - name: # [string] Name - Parameter name.
    value: # [string] Value - JavaScript expression to compute the parameter
authRequestHeaders: # [array] Authentication headers - Optional authenticatio
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header's v

# -------------------------------------------------------


# -------------- if authentication is oauthSecret ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
clientSecretParamName: # [string] Client secret parameter - Parameter name th
textSecret: # [string] Client secret value (text secret) - Select (or create)
```

```
  authRequestParams: # [array] Extra authentication parameters - OAuth request
    - name: # [string] Name - Parameter name.
      value: # [string] Value - JavaScript expression to compute the parameter
  authRequestHeaders: # [array] Authentication headers - Optional authenticatio
    - name: # [string] Name - Header name.
      value: # [string] Value - JavaScript expression to compute the header's v

  # --------------------------------------------------------

  timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout,
  defaultBreakers: # [string] Hidden Default Breakers
  safeHeaders: # [array of strings] Safe headers - List of headers that are saf

# --------------------------------------------------------



# -------------- if type is office365_mgmt ---------------

conf: # [object] [required]
  plan_type: # [string] Subscription plan - Office 365 subscription plan for yo
  tenant_id: # [string] [required] Tenant identifier - Directory ID (tenant ide
  app_id: # [string] [required] Application identifier - Identifier of the regi
  client_secret: # [string] [required] Client secret - Application key of the r
  publisher_identifier: # [string] Publisher identifier - Optional PublisherIde
  content_type: # [string] [required] Content type - The type of content to ret

# --------------------------------------------------------



# -------------- if type is office365_service ---------------

conf: # [object] [required]
  tenant_id: # [string] Tenant identifier - Directory ID (tenant identifier) in
  app_id: # [string] [required] Application identifier - Identifier of the regi
  client_secret: # [string] [required] Client secret - Application key of the r
  content_type: # [string] [required] Content type - The type of content to ret

# --------------------------------------------------------



# -------------- if type is prometheus ---------------

conf: # [object] [required]
```

```
  dimensionList: # [array] Extra Dimensions - Other dimensions to include in ev
  username: # [string] Username - Optional username for Basic authentication
  password: # [string] Password - Optional password for Basic authentication
  discoveryType: # [string] Discovery Type - Target discovery mechanism, use st

  # -------------- if discoveryType is static ---------------

  targetList: # [array] Targets - List of Prometheus targets to pull metrics fr

  # --------------------------------------------------------


  # -------------- 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
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics fro

  # --------------------------------------------------------


  # -------------- if discoveryType is ec2 ---------------

  usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
  scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting
  scrapePort: # [number] Metrics Port - The port number in the metrics URL for
  scrapePath: # [string] Metrics Path - Path to use when collecting metrics fro
  searchFilter: # [array] Search Filter - EC2 Instance Search Filter
    - Name: # [string] Filter Name - Search filter attribute name, see: https:/
      Values: # [array of strings] Filter Values - Search Filter Values
  region: # [string] Region - Region from which to retrieve data.
  awsAuthenticationMethod: # [string] Authentication Method - AWS authenticatio
  enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credential
  endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to A
  signatureVersion: # [string] Signature version - Signature version to use for

  # --------------------------------------------------------


# --------------------------------------------------------
# -------------- if type is rest ---------------

conf: # [object] [required]
  discovery: # [object]
    discoverType: # [string] Discover type - Defines how task discovery will be

    # -------------- if discoverType is http ---------------

    discoverUrl: # [string] Discover URL - Expression to derive URL to use for
    discoverMethod: # [string] Discover method - Discover HTTP method.
    discoverRequestHeaders: # [array] Discover headers - Optional discover requ
      - name: # [string] Name - Header name.
        value: # [string] Value - JavaScript expression to compute the paramete
    discoverDataField: # [string] Discover data field - Path to field in the re

    # ----------------------------------------------------------


    # -------------- if discoverType is json ---------------

    manualDiscoverResult: # [string] Discover result - Allows hard-coding the D
    discoverDataField: # [string] Discover data field - Within the response JSO

    # ----------------------------------------------------------


    # -------------- if discoverType is list ---------------

    itemList: # [array of strings] Discover items - Comma-separated list of ite

    # ----------------------------------------------------------

  collectUrl: # [string] Collect URL - Expression to derive URL to use for the
  collectMethod: # [string] [required] Collect method - Collect HTTP method.

  # -------------- if collectMethod is get ---------------

  collectRequestParams: # [array] Collect parameters - Optional collect request
    - name: # [string] Name - Parameter name
      value: # [string] Value - JavaScript expression to compute the parameter

  # ----------------------------------------------------------
  # -------------- if collectMethod is post ---------------

  collectRequestParams: # [array] Collect parameters - Optional collect request
    - name: # [string] Name - Parameter name.
      value: # [string] Value - JavaScript expression to compute the parameter

  # --------------------------------------------------------



  # -------------- if collectMethod is post_with_body ---------------

  collectBody: # [string] Collect POST body - Template for POST body to send wi

  # --------------------------------------------------------



  # -------------- if collectMethod is other ---------------

  collectVerb: # [string] Collect verb - DIY HTTP verb to use for the collect o
  collectBody: # [string] Collect body - Template for body to send with the Col
  collectRequestParams: # [array] Collect parameters - Optional collect request
    - name: # [string] Name - Parameter name.
      value: # [string] Value - JavaScript expression to compute the parameter

  # --------------------------------------------------------

  collectRequestHeaders: # [array] Collect headers - Optional collect request h
    - name: # [string] Name - Header Name
      value: # [string] Value - JavaScript expression to compute the header's v
  pagination: # [object]
    type: # [string] Pagination - Select collect pagination scheme

    # -------------- if type is response_body ---------------

    attribute: # [array,string] Response attribute - The name of the attribute
    maxPages: # [number] Max pages - The maximum number of pages to retrieve, s

    # --------------------------------------------------------



    # -------------- if type is response_header_link ---------------
```

```
    nextRelationAttribute: # [string] Next page relation name - Relation name u
    curRelationAttribute: # [string] Current page relation name - Optional rela
    maxPages: # [number] Max pages - The maximum number of pages to retrieve, s

    # ---------------------------------------------------------



    # -------------- if type is request_offset ---------------

    offsetField: # [string] Offset field name - Query string parameter that set
    offset: # [number] Starting offset - Offset index from which to start reque
    offsetSpacer: # [null]
    limitField: # [string] Limit field name - Query string parameter to set the
    limit: # [number] Limit - Maximum number of records to collect per request.
    limitSpacer: # [null]
    totalRecordField: # [string] Total record count field name - Identifies the
    maxPages: # [number] Max pages - The maximum number of pages to retrieve. S
    zeroIndexed: # [boolean] Zero-based index - Toggle to Yes to indicate that

    # ---------------------------------------------------------



    # -------------- if type is request_page ---------------

    pageField: # [string] Page number field name - Query string parameter that
    page: # [number] Starting page number - Page number from which to start req
    offsetSpacer: # [null]
    sizeField: # [string] Page size field name - Query string parameter to set
    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 at
    totalRecordField: # [string] Total record count field name - Identifies the
    maxPages: # [number] Max pages - The maximum number of pages to retrieve. S
    zeroIndexed: # [boolean] zero-based index - Toggle to Yes to indicate that

    # ---------------------------------------------------------

  authentication: # [string] [required] Authentication - Authentication method

  # -------------- 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 store

# -------------------------------------------------------


# -------------- if authentication is login ---------------

loginUrl: # [string] Login URL - URL to use for login API call. This call is
username: # [string] Username - Login username
password: # [string] Password - Login password
loginBody: # [string] POST body - Template for POST body to send with login r
tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
authRequestHeaders: # [array] Authentication headers - Optional authenticatio
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the header's v

# -------------------------------------------------------


# -------------- if authentication is loginSecret ---------------

loginUrl: # [string] Login URL - URL to use for login API call, this call is
credentialsSecret: # [string] Credentials secret - Select (or create) a store
loginBody: # [string] POST body - Template for POST body to send with login r
tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
authRequestHeaders: # [array] Authentication headers - Optional authenticatio
  - name: # [string] Name - Header name.
    value: # [string] Value - JavaScript expression to compute the parameter

# -------------------------------------------------------


# -------------- if authentication is oauth ---------------

loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
```

```
    tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
    authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
    clientSecretParamName: # [string] Client secret parameter - Parameter name th
    clientSecretParamValue: # [string] Client secret value - Secret value to add
    authRequestParams: # [array] Extra authentication parameters - OAuth request
      - name: # [string] Name - Parameter name.
        value: # [string] Value - JavaScript expression to compute the parameter
    authRequestHeaders: # [array] Authentication headers - Optional authenticatio
      - name: # [string] Name - Header name.
        value: # [string] Value - JavaScript expression to compute the header's v

    # --------------------------------------------------------



    # -------------- if authentication is oauthSecret ---------------

    loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
    tokenRespAttribute: # [string] Token attribute - Path to token attribute in l
    authHeaderExpr: # [string] Authorize expression - JavaScript expression to co
    clientSecretParamName: # [string] Client secret parameter - Parameter name th
    textSecret: # [string] Client secret value (text secret) - Select (or create)
    authRequestParams: # [array] Extra authentication parameters - OAuth request
      - name: # [string] Name - Parameter name.
        value: # [string] Value - JavaScript expression to compute the parameter
    authRequestHeaders: # [array] Authentication headers - Optional authenticatio
      - name: # [string] Name - Header name.
        value: # [string] Value - JavaScript expression to compute the header's v

    # --------------------------------------------------------

    timeout: # [number] Request timeout (secs) - HTTP request inactivity timeout,
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
    disableTimeFilter: # [boolean] Disable time filter - Used to disable collecto
    safeHeaders: # [array of strings] Safe headers - List of headers that are saf

  # --------------------------------------------------------



  # -------------- if type is s3 ---------------

  conf: # [object] [required]
    outputName: # [string] Auto-populate from - The name of the predefined Destin
    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
    extractors: # [array] Path extractors - Allows using template tokens as conte
      - key: # [string] Token - A token from the template path, e.g.: epoch
        expression: # [string] Extractor expression - JS expression that receive
    awsAuthenticationMethod: # [string] Authentication Method - AWS authenticatio

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key. If not present, will fall bac
    awsSecretKey: # [string] Secret key - Secret key. If not present, will fall 

    # --------------------------------------------------------



    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret th

    # --------------------------------------------------------

    endpoint: # [string] Endpoint - S3 service endpoint. If empty, the endpoint w
    signatureVersion: # [string] Signature version - Signature version to use fo
    enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credentia
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
    assumeRoleExternalId: # [string] External ID - External ID to use when assum
    recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
    maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metada
    reuseConnections: # [boolean] Reuse Connections - Whether to reuse connectio
    rejectUnauthorized: # [boolean] Reject Unauthorized Certificates - Whether t
    verifyPermissions: # [boolean] Verify bucket permissions - Disable if you ca

  # --------------------------------------------------------



  # -------------- if type is script ---------------

  conf: # [object] [required]
    discoverScript: # [string] Discover Script - Script to discover what to coll
    collectScript: # [string] [required] Collect Script - Script to run to perfo
    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
    search: # [string] Search - Enter Splunk search here. For example: 'index=my/
    earliest: # [string] Earliest - The earliest time boundary for the search. Ca
    latest: # [string] Latest - The latest time boundary for the search. Can be a
    endpoint: # [string] [required] Search endpoint - REST API used to create a s
    outputMode: # [string] [required] Output mode - Format of the returned output
    collectRequestParams: # [array] Extra parameters - Optional collect request p
      - name: # [string] Name - Parameter name
        value: # [string] Value - JavaScript expression to compute the parameter
    collectRequestHeaders: # [array] Extra headers - Optional collect request hea
      - name: # [string] Name - Header Name
        value: # [string] Value - JavaScript expression to compute the header's v
    authentication: # [string] [required] Authentication - Authentication method

    # -------------- 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 store

    # --------------------------------------------------------


    # -------------- if authentication is login ---------------

    loginUrl: # [string] Login URL - URL to use for login API call. This call is
    username: # [string] Username - Login username
    password: # [string] Password - Login password
    loginBody: # [string] POST Body - Template for POST body to send with login r
    tokenRespAttribute: # [string] Token Attribute - Path to token attribute in l
    authHeaderExpr: # [string] Authorize Expression - JavaScript expression to co
```

```
    # ---------------------------------------------------------


    # -------------- if authentication is loginSecret ---------------

    loginUrl: # [string] Login URL - URL to use for login API call, this call is
    credentialsSecret: # [string] Credentials secret - Select (or create) a store
    loginBody: # [string] POST Body - Template for POST body to send with login r
    tokenRespAttribute: # [string] Token Attribute - Path to token attribute in l
    authHeaderExpr: # [string] Authorize Expression - JavaScript expression to co

    # ---------------------------------------------------------

    timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout,
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
    disableTimeFilter: # [boolean] Disable time filter - Used to disable collecto

  # ---------------------------------------------------------

  destructive: # [boolean] Destructive - If set to Yes, the collector will delete
input: # [object]
  type: # [string]
  breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
  staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of ti
  sendToRoutes: # [boolean] Send to Routes - If set to Yes, events will be sent t

  # -------------- if sendToRoutes is true ---------------

  pipeline: # [string] Pre-Processing Pipeline - Pipeline to process results befo

  # ---------------------------------------------------------


  # -------------- 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) an
    command: # [string] Command - Command to feed the data through (via stdin) a
    args: # [array of strings] Arguments - Arguments
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thro
  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, 
type: # [string] Job type - Job type.
ttl: # [string] Time to live - Time to keep the job's artifacts on disk after jo
removeFields: # [array of strings] Remove Discover fields - List of fields to re
resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failu
environment: # [string] Environment - Optionally, enable this config only on a s
schedule: # [object] Schedule - Configuration for a scheduled job.
  enabled: # [boolean] Enabled - Determines whether or not this schedule is enab

  # -------------- if enabled is true ---------------

  cronSchedule: # [string] Cron schedule - A cron schedule on which to run this 
  maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of inst
  skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their 
  run: # [object] Run settings

    # -------------- if type is collection ---------------

    rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that 
    maxTaskReschedule: # [number] Max task reschedule - Max number of times a ta
    logLevel: # [string] Log Level - Level at which to set task logging.
    jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run 
    mode: # [string] Mode - Job run mode. Preview will either return up to N mat

    # -------------- if mode is list ---------------

    discoverToRoutes: # [boolean] Send to Routes - If true, send discover result

    # -------------------------------------------------------



    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture Time (sec) - Amount of time to keep capture o
      maxEvents: # [number] Capture Up to N Events - Maximum number of events to
      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 a
      expression: # [string] Filter - A filter for tokens in the provided collect p
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for s
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for t

    # -------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in St
executor_job: # [object]
  executor: # [object]
    type: # [string] Executor type - The type of executor to run.
    storeTaskResults: # [boolean] Store task results - Determines whether or not to
    conf: # [object] Executor-specific settings.
  type: # [string] Job type - Job type.
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
  removeFields: # [array of strings] Remove Discover fields - List of fields to rem
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failur
  environment: # [string] Environment - Optionally, enable this config only on a sp
  schedule: # [object] Schedule - Configuration for a scheduled job.
    enabled: # [boolean] Enabled - Determines whether or not this schedule is enabl

    # -------------- if enabled is true ---------------

    cronSchedule: # [string] Cron schedule - A cron schedule on which to run this j
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of insta
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their r
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a tas
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run (
```

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 Worker Node.

```
      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
```

```
    discoverToRoutes: # [boolean] Send to Routes - If true, send discover result

    # --------------------------------------------------------

    # -------------- if mode is preview ---------------

    capture: # [object] Capture Settings
      duration: # [number] Capture Time (sec) - Amount of time to keep capture o
      maxEvents: # [number] Capture Up to N Events - Maximum number of events to
      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
    expression: # [string] Filter - A filter for tokens in the provided collect
    minTaskSize: # [string] Lower task bundle size - Limits the bundle size for
    maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for

  # --------------------------------------------------------

streamtags: # [array of strings] Tags - Add tags for filtering and grouping in S
```

19.6.8. Job-Limits.Yml

`job-limits.yml` maintains parameters for Collection jobs and system tasks. In the UI, you can configure these per Worker Group at **Group Settings** > **General Settings** > **Limits** > **Jobs**.

Where we do not specify or validate a minimum or maximum value, Cribl provides a prudent default. Adjust this initial value to match your architecture and needs.

$CRIBL_HOME/default/cribl/job-limits.yml

```
concurrentJobLimit: # [number; default: 10, minimum: 1] Concurrent job limit - The
concurrentSystemJobLimit: # [number; default: 10, minimum: 1] Concurrent system jo
concurrentScheduledJobLimit: # [number, default: -2] Concurrent scheduled job limi
concurrentTaskLimit: # [number; default: 2, minimum: 1] Concurrent task limit - Th
concurrentSystemTaskLimit: # [number; default: 1, minimum: 1] Concurrent system ta
maxTaskPerc: # [number; default: 0.5, minimum: 0, maximum: 1] Max task usage perce
taskPollTimeoutMs: # [number; default: 60000, minimum: 10000] Task poll timeout -
jobArtifactsReaperPeriod: # [string; default: 30 min.] Artifact reaper period - Ti
finishedJobArtifactsLimit: # [number; default: 100 items, minimum: 0] Finished job
finishedTaskArtifactsLimit: # [number; default: 500 items, minimum: 0] Finished ta
taskManifestFlushPeriodMs: # [number; default: 100, minimum: 100, maximum: 10000]
taskManifestMaxBufferSize: # [number; default: 1000, minimum: 100, maximum: 10000]
taskManifestReadBufferSize: # [string; default: 4 KB] Manifest reader buffer size
schedulingPolicy: # [string; default: LeastInFlight] Job dispatching - The method
jobTimeout: # [string; default: 0] Job timeout - Maximum time a job is allowed to
taskHeartbeatPeriod: # [number; default: 60, minimum: 60] Task heartbeat period -
```

19.6.9. Licenses.Yml

`licenses.yml` maintains a list of Cribl Stream licenses.

$CRIBL_HOME/default/cribl/licenses.yml

```
licenses: # [array of string] - list of licenses
```

19.6.10. 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/Fleet-name>` > **Group Settings** > **General Settings** > **Limits**.

Where we do not specify or validate a minimum or maximum value, Cribl provides a prudent default. Adjust this initial value to match your architecture and needs.

$CRIBL_HOME/default/cribl/limits.yml

```
samples: # [object] Samples
  maxSize: # [string; maximum: 3 MB; default: 256 KB] Max sample size - Maximum fil
maxMetrics: # [number; default: 1000000] Total allowed number of metric series, con
minFreeSpace: # [string; default: 5 GB] Min free disk space - The minimum amount of
maxPQSize: # [string; default: 1 TB] Maximum persistent queue size per Worker Proce
metricsGCPeriod: # [string; default: 60s] Metrics GC period - The interval on which
metricsMaxCardinality: # [number; default: 1000] Metrics cardinality limit - The sy
metricsMaxDiskSpace: # [string; default: 64GB] Metrics max disk space - Maximum all
metricsWorkerIdBlacklist: # [array of strings] Metrics worker tracking - List of me
metricsNeverDropList: # [array of strings] Metrics never-drop list - List of metric
metricsFieldsBlacklist: # [array of strings] Disable field metrics - List of event
metricsDirectory: # [string] Metrics directory - Directory in which to store metric
cpuProfileTTL: # [string; default: 30m] CPU profile TTL - The time-to-live for coll
eventsMetadataSources: # [array of strings] Event metadata sources - List of event
edgeMetricsMode: # [string] Metrics to send from Edge Nodes - Choose a set of metri

# -------------- if edgeMetricsMode is custom ---------------

edgeMetricsCustomExpression: # [string] Metrics expression - JavaScript expression

# -------------------------------------------------------

enableMetricsPersistence: # [boolean; default: true] Persist metrics - Set this to
```

19.6.11. 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: erro
  DEFAULT: info
  input:DistMaster: debug
  output:DistWorker: debug
```

19.6.12. 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]
```

19.6.13. 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]
```

19.6.14. Outputs.Yml

`outputs.yml` contains configuration settings for Cribl Stream 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
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatica
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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
    format: # [string] Format - Specifies how to format events before sending out.

    # -------------- if format is custom ---------------

    customSourceExpression: # [string] Source expression - Expression to evaluate
    customDropWhenNull: # [boolean] Drop when null - Whether or not to drop events
    customEventDelimiter: # [string] Event delimiter - Delimiter string to insert
    customContentType: # [string] Content type - Content type to use for request.

    # --------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing reques
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
    maxPayloadEvents: # [number] Max events per request - Max number of events to
    compress: # [boolean] Compress - Whether to compress the payload body before s
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each qu
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the qu
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # -------------------------------------------------------

    authType: # [string] Authentication type - The authentication method to use for

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # -------------------------------------------------------



    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # -------------------------------------------------------



    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret

    # -------------------------------------------------------



    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text s

    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  devnull_output: # [object]
    type: # [string] Output Type
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  syslog_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Protocol - The network protocol to use for sending out sys

    # -------------- if protocol is tcp ---------------

    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
      certificateName: # [string] Certificate name - The name of the predefined cer
      caPath: # [string] CA certificate path - Path on client in which to find CA c
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

      # --------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # --------------------------------------------------------


    # -------------- 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

    # --------------------------------------------------------
    facility: # [number] Facility - Default value for message facility, will be ove
    severity: # [number] Severity - Default value for message severity, will be ove
    appName: # [string] App Name - Default value for application name, will be over
    messageFormat: # [string] Message Format - The syslog message format depending
    timestampFormat: # [string] Timestamp Format - Timestamp format to use when ser
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throt
    octetCountFraming: # [boolean] Octet count framing - When enabled, messages wil
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 serializ
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throt
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
      certificateName: # [string] Certificate name - The name of the predefined cer
      caPath: # [string] CA certificate path - Path on client in which to find CA c
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

      # --------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in mul
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shutt
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
    authToken: # [string] Auth token - Shared secret token to use when establishing

    # -------------- if authType is manual ---------------



    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

    # -------------- if authType is secret ---------------



    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  splunk_lb_output: # [object]
    type: # [string] Output Type
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
    nestedFields: # [string] Nested field serialization - Specifies how to serializ
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thrott
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
      certificateName: # [string] Certificate name - The name of the predefined cer
      caPath: # [string] CA certificate path - Path on client in which to find CA c
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

      # ----------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in mul
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is shutt
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # ----------------------------------------------------------

    indexerDiscovery: # [boolean] Indexer Discovery - Automatically discover indexe

    # -------------- if indexerDiscovery is true ---------------

    indexerDiscoveryConfigs: # [object]  - List of configurations to set up indexer
      site: # [string] [required] Site - Clustering site of the indexers from where
      masterUri: # [string] Cluster Manager URI - Full URI of Splunk cluster Manage
      refreshIntervalSec: # [number] [required] Refresh Period - Time interval in s
      authType: # [string] Authentication method - Enter a token directly, or provi
      authToken: # [string] Auth token - Authentication token required to authentic

      # -------------- if authType is manual ---------------


      # ----------------------------------------------------------
      textSecret: # [string] Auth token (text secret) - Select (or create) a storec

      # -------------- if authType is secret ---------------


      # -------------------------------------------------------


    # -------------------------------------------------------


    # -------------- if indexerDiscovery is false ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curr
    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
        servername: # [string] TLS Servername - Servername to use if establishing a
        weight: # [number] Load Weight - The weight to use for load-balancing purpo
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

    # -------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
    authToken: # [string] Auth token - Shared secret token to use when establishing

    # -------------- if authType is manual ---------------


    # -------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

    # -------------- if authType is secret ---------------


    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
```

```
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 eve
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS

  # -------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the cur
  urls: # [array] Splunk HEC Endpoints
    - url: # [string] HEC Endpoint - URL to a Splunk HEC endpoint to send events
      weight: # [number] Load Weight - The weight to use for load-balancing purp
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve a
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - Ho

  # -------------------------------------------------------

  nextQueue: # [string] Next Processing Queue - Which Splunk processing queue to
  tcpRouting: # [string] Default _TCP_ROUTING - Set the value of _TCP_ROUTING fo
  concurrency: # [number] Request concurrency - Maximum number of ongoing reques
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
  maxPayloadEvents: # [number] Max events per request - Max number of events to
  compress: # [boolean] Compress - Whether to compress the payload body before s
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  enableMultiMetrics: # [boolean] Output multi-metrics - Output metrics in multi
  authType: # [string] Authentication method - Enter a token directly, or provid

  # -------------- 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 stor

  # ----------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 curr
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
    servername: # [string] TLS Servername - Servername to use if establishing a
    weight: # [number] Load Weight - The weight to use for load-balancing purpo
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

# ----------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throt
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ----------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
  certificateName: # [string] Certificate name - The name of the predefined cer
  caPath: # [string] CA certificate path - Path on client in which to find CA c
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
  certPath: # [string] Certificate path (mutual auth) - Path on client in which
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

  # ----------------------------------------------------------

connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befo
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

# -------------- if onBackpressure is queue ----------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
```

```
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ----------------------------------------------------------

  authType: # [string] Authentication method - Enter a token directly, or provide
  authToken: # [string] Auth token - Optional authentication token to include as

  # -------------- if authType is manual ----------------



  # ----------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

  # -------------- if authType is secret ----------------



  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
wavefront_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Enter a token directly, or provide

  # -------------- if authType is manual ----------------

  token: # [string] Auth token - WaveFront API authentication token (see [here](h

  # ----------------------------------------------------------



  # -------------- if authType is secret ----------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

  # ----------------------------------------------------------

  domain: # [string] Domain name - WaveFront domain name, e.g. "longboard"
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 l
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
signalfx_output: # [object]
  type: # [string] Output Type
  authType: # [string] Authentication method - Enter a token directly, or provide

  # -------------- if authType is manual ---------------

  token: # [string] Auth token - SignalFx API access token (see [here](https://do

  # ---------------------------------------------------------



  # -------------- if authType is secret ---------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t
  # -----------------------------------------------------------

  realm: # [string] Realm - SignalFx realm name, e.g. "us0"
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are r
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 l
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # -----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
filesystem_output: # [object]
  type: # [string] Output Type
  destPath: # [string] Output location - Final destination for the output files
  stagePath: # [string] Staging location - Filesystem location in which to buffer
  addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loc
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging directo

  # -------------- if removeEmptyDirs is true ---------------
  emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to cle

  # ----------------------------------------------------------

  partitionExpr: # [string] Partitioning expression - JS expression defining how
  format: # [string] Data format - Format of the output data.

  # -------------- if format is json ---------------

  compress: # [string] Compress - Choose data compression format to apply before

  # ----------------------------------------------------------



  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row grou
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are su
  parquetDataPageVersion: # [string] Data page version - Serialization format of
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique row

  # ----------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of tin
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of tin
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop eve
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
s3_output: # [object]
  type: # [string] Output Type
  bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be
  region: # [string] Region - Region where the S3 bucket is located.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication
```

```
# -------------- 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 tha

# --------------------------------------------------------

awsSecretKey: # [string] Secret key - Secret key
endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS
signatureVersion: # [string] Signature version - Signature version to use for
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to a
assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the r
assumeRoleExternalId: # [string] External ID - External ID to use when assumin
stagePath: # [string] [required] Staging location - Filesystem location in whi
addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging lo
removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging direct

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to cl

# --------------------------------------------------------

destPath: # [string] [required] Key prefix - Prefix to append to files before
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
storageClass: # [string] Storage class - Storage class to select for uploaded
serverSideEncryption: # [string] Server side encryption - Server-side encrypti

# -------------- if serverSideEncryption is aws:kms ---------------

kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer managed key to

# --------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how
```

```
  partitionExpr: # [string] Partitioning expression - JS expression defining how
  format: # [string] Data format - Format of the output data.

  # -------------- if format is json ---------------

  compress: # [string] Compress - Choose data compression format to apply before

  # -------------------------------------------------------

  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schema
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row grou
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are s
  parquetDataPageVersion: # [string] Data page version - Serialization format of
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique ro

  # -------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression t
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output fil
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of ti
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of ti
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop ev
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatica
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
azure_blob_output: # [object]
  type: # [string] Output Type
  containerName: # [string] Container name - A container organizes a set of blob
  createContainer: # [boolean] Create container - Creates the configured contain
  destPath: # [string] [required] Blob prefix - Root directory prepended to path
  stagePath: # [string] [required] Staging location - Filesystem location in whi
  addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging lo
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging direct

  # -------------- if removeEmptyDirs is true ---------------
```

```
emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to cle

# ----------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how
format: # [string] Data format - Format of the output data.

# -------------- if format is json ---------------

compress: # [string] Compress - Choose data compression format to apply before

# ----------------------------------------------------------



# -------------- if format is parquet ---------------

parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
parquetRowGroupSize: # [string] Row group size - Ideal memory size for row grou
parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.
spacer: # [null]
parquetVersion: # [string] Parquet version - Determines which data types are su
parquetDataPageVersion: # [string] Data page version - Serialization format of
shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique row

# ----------------------------------------------------------

baseFileName: # [string] File name prefix expression - JavaScript expression to
fileNameSuffix: # [string] File name suffix expression - JavaScript expression
maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of tim
maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of tim
maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
onBackpressure: # [string] Backpressure behavior - Whether to block or drop eve
authType: # [string] Authentication method - Enter connection string directly,
connectionString: # [string] Connection string - Enter your Azure Storage accou

# -------------- if authType is manual ---------------



# ----------------------------------------------------------

textSecret: # [string] Connection string (text secret) - Select (or create) a s
```

```
  # -------------- if authType is secret ---------------


  # ---------------------------------------------------------


  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
azure_logs_output: # [object]
  type: # [string] Output Type
  logType: # [string] Log Type - The Log Type of events sent to this LogAnalytics
  resourceId: # [string] Resource ID - Optional Resource ID of the Azure resource
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 l
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ---------------------------------------------------------

  authType: # [string] Authentication method - Enter workspace ID and workspace k
  workspaceId: # [string] Workspace ID - Azure Log Analytics Workspace ID. See Az
  workspaceKey: # [string] Workspace key - Azure Log Analytics Workspace Primary
```

```
    # -------------- if authType is manual ---------------


    # -------------------------------------------------------

    keypairSecret: # [string] Secret key pair - Select (or create) a stored secret

    # -------------- if authType is secret ---------------


    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  kinesis_output: # [object]
    type: # [string] Output Type
    streamName: # [string] Stream Name - Kinesis stream name to send events to.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication

    # -------------- 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

    # -------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is locat
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defa
    signatureVersion: # [string] Signature version - Signature version to use for s
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role crede
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
    concurrency: # [number] Put request concurrency - Maximum number of ongoing put
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  honeycomb_output: # [object]
    type: # [string] Output Type
    dataset: # [string] Dataset name - Name of the dataset to send events to â e.
    concurrency: # [number] Request concurrency - Maximum number of ongoing request
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
    maxPayloadEvents: # [number] Max events per request - Max number of events to i
    compress: # [boolean] Compress - Whether to compress the payload body before se
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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 l
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue fil
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # -----------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select
    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

    # -------------- if authType is secret ---------------



    # -----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  azure_eventhub_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to cor
    topic: # [string] [required] Event Hub Name - The name of the Event Hub (a.k.a.
    ack: # [number] Acknowledgments - Control the number of required acknowledgment
    format: # [string] Record data format - Format to use to serialize events befor
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (
    flushEventCount: # [number] Max events per batch - Maximum number of events in
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
    sasl: # [object] Authentication - Authentication parameters to use when connect
      disabled: # [boolean] Disabled - Enable authentication.

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
      username: # [string] Username - The username for authentication. For Event Hu
      username: # [string] Username - The username for authentication. For Event Hu
      authType: # [string] Authentication method - Enter password directly, or sele

      # --------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this

      # --------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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 te

    # -------------------------------------------------------

    logType: # [string] Log type - Log type value to send to Chronicle. Can be over
    logTextField: # [string] Log text field - Name of the event field that contains
    logFormatType: # [string] Send events as
    region: # [string] Region - Regional endpoint to send events to
    concurrency: # [number] Request concurrency - Maximum number of ongoing request
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
    maxPayloadEvents: # [number] Max events per request - Max number of events to i
    compress: # [boolean] Compress - Whether to compress the payload body before se
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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 l
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  google_cloud_storage_output: # [object]
    type: # [string] Output Type
    bucket: # [string] Bucket name - Name of the destination Bucket. This value can
    region: # [string] [required] Region - Region where the bucket is located.
    [illegible]
```

```
endpoint: # [string] [required] Endpoint - Google Cloud Storage service endpoir
signatureVersion: # [string] Signature version - Signature version to use for s
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

# --------------------------------------------------------

stagePath: # [string] [required] Staging location - Filesystem location in whic
destPath: # [string] [required] Key prefix - Prefix to append to files before u
objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
storageClass: # [string] Storage class - Storage class to select for uploaded o
reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loc
removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging directo

# -------------- if removeEmptyDirs is true ---------------

emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to cle

# --------------------------------------------------------

partitionExpr: # [string] Partitioning expression - JS expression defining how
format: # [string] Data format - Format of the output data.

# -------------- if format is json ---------------

compress: # [string] Compress - Choose data compression format to apply before

# --------------------------------------------------------
```

```
  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row grou
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are su
  parquetDataPageVersion: # [string] Data page version - Serialization format of
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique row

  # -------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of tin
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of tin
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop eve
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
google_pubsub_output: # [object]
  type: # [string] Output Type
  topicName: # [string] Topic ID - ID of the topic to send events to.
  createTopic: # [boolean] Create topic - If enabled, create topic if it does not
  orderedDelivery: # [boolean] Ordered delivery - If enabled, send events in the
  region: # [string] Region - Region to publish messages to. Select 'default' to
  googleAuthMethod: # [string] Authentication Method - Google authentication meth

  # -------------- if googleAuthMethod is manual ---------------

  serviceAccountCredentials: # [string] Service account credentials - Contents of

  # -------------------------------------------------------



  # -------------- if googleAuthMethod is secret ---------------

  secret: # [string] Service account credentials (text secret) - Select (or creat

  # -------------------------------------------------------
  batchSize: # [number] Batch size - The maximum number of items the Google API s
  batchTimeout: # [number] Batch timeout (ms) - The maximum amount of time, in mi
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches befo
  maxRecordSizeKB: # [number] Max batch size (KB) - Maximum size (KB) of batches
  maxInProgress: # [number] Max concurrent requests - The maximum number of in-pr
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
kafka_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Kafka brokers to connect to, e.
  topic: # [string] [required] Topic - The topic to publish events to. Can be ove
  ack: # [number] Acknowledgments - Control the number of required acknowledgment
  format: # [string] Record data format - Format to use to serialize events befor
  compression: # [string] Compression - Codec to use to compress the data before
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (
  flushEventCount: # [number] Max events per batch - Maximum number of events in
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 Con
    defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaI
    defaultValueSchemaId: # [number] Default value schema ID - Used when __valueS
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled
      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that a
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
      certificateName: # [string] Certificate name - The name of the predefined c
      caPath: # [string] CA certificate path - Path on client in which to find CA
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
      certPath: # [string] Certificate path (mutual auth) - Path on client in whi
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private ke
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use whe
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use whe

      # --------------------------------------------------------


    # --------------------------------------------------------

  connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
  requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
  sasl: # [object] Authentication - Authentication parameters to use when connect
    disabled: # [boolean] Disabled - Enable Authentication

    # -------------- if disabled is false ---------------

    mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

    # --------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined cer
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
```

```
    # -------------------------------------------------------

  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # -------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
confluent_cloud_output: # [object]
  type: # [string] Output Type
  brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to conn
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined cer
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # -------------------------------------------------------

  topic: # [string] [required] Topic - The topic to publish events to. Can be ove
  ack: # [number] Acknowledgments - Control the number of required acknowledgment
  format: # [string] Record data format - Format to use to serialize events befor
```

```
compression: # [string] Compression - Codec to use to compress the data before
maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (
flushEventCount: # [number] Max events per batch - Maximum number of events in
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
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 Cor
  defaultKeySchemaId: # [number] Default key schema ID - Used when __keySchemaI
  defaultValueSchemaId: # [number] Default value schema ID - Used when __valueS
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that a
    servername: # [string] Server name (SNI) - Server name for the SNI (Server
    certificateName: # [string] Certificate name - The name of the predefined c
    caPath: # [string] CA certificate path - Path on client in which to find CA
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in
    certPath: # [string] Certificate path (mutual auth) - Path on client in whi
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private ke
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use whe
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use whe

    # ---------------------------------------------------------


  # ---------------------------------------------------------

connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait fo
requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a su
sasl: # [object] Authentication - Authentication parameters to use when connect
  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 q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS l

    # ---------------------------------------------------------



    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the curr
    urls: # [array] Bulk API URLs
      - url: # [string] URL - URL to an Elastic node to send events to â e.g., ht
        weight: # [number] Load Weight - The weight to use for load-balancing purpo
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How

    # ---------------------------------------------------------

    index: # [string] Index or Data Stream - Index or Data Stream to send events to
    docType: # [string] Type - Document type to use for events. Can be overwritten
    concurrency: # [number] Request concurrency - Maximum number of ongoing request
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
    maxPayloadEvents: # [number] Max events per request - Max number of events to i
    compress: # [boolean] Compress - Whether to compress the payload body before se
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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 w
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    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 s

      # ---------------------------------------------------------

    elasticVersion: # [string] Elastic Version - Optional Elasticsearch version, us
    elasticPipeline: # [string] Elastic pipeline - Optional Elasticsearch destinati
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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.: c
messageField: # [string] Log message field - Name of field to send as log messa
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, e
concurrency: # [number] Request concurrency - Maximum number of ongoing request
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
maxPayloadEvents: # [number] Max events per request - Max number of events to i
compress: # [boolean] Compress - Whether to compress the payload body before se
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
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 l
failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
pqControls: # [object]

# ---------------------------------------------------------

authType: # [string] Authentication method - Enter API key directly, or select
apiKey: # [string] API key - New Relic API key. Can be overridden using __newRe

# -------------- if authType is manual ---------------



# ---------------------------------------------------------

textSecret: # [string] API key (text secret) - Select (or create) a stored text
```

```
  # -------------- if authType is secret ---------------


  # -------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 no
  concurrency: # [number] Request concurrency - Maximum number of ongoing reques
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
  maxPayloadEvents: # [number] Max events per request - Max number of events to
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each qu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the qu
  pqPath: # [string] Queue file path - The location for the persistent queue fil
  pqCompress: # [string] Compression - Codec to use to compress the persisted da
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop ev
  pqControls: # [object]

  # -------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select
  apiKey: # [string] API key - New Relic API key. Can be overridden using __newR
  # -------------- if authType is manual ---------------


  # ----------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select (or create) a stored text

  # -------------- if authType is secret ---------------


  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
influxdb_output: # [object]
  type: # [string] Output Type
  url: # [string] Write API URL - URL of an InfluxDB cluster to send events to, e
  useV2API: # [boolean] Use v2 API - The v2 API can be enabled with InfluxDB vers

  # -------------- if useV2API is false ---------------

  database: # [string] Database - Database to write to.

  # ----------------------------------------------------------


  # -------------- if useV2API is true ---------------

  bucket: # [string] Bucket - Bucket to write to.
  org: # [string] Organization - Organization ID for this bucket.

  # ----------------------------------------------------------

  timestampPrecision: # [string] Timestamp precision - Sets the precision for the
  dynamicValueFieldName: # [boolean] Dynamic value fields - Enabling this will pu
  valueFieldName: # [string] Value field name - Name of the field in which to sto
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
```

```
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
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 l
failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
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

# ---------------------------------------------------------
```

```
  # -------------- if authType is textSecret ---------------

  textSecret: # [string] Token (text secret) - Select (or create) a stored text s

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
cloudwatch_output: # [object]
  type: # [string] Output Type
  logGroupName: # [string] Log group name - CloudWatch log group to associate eve
  logStreamName: # [string] [required] Log stream prefix - Prefix for CloudWatch
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication

  # -------------- 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

  # --------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] [required] Region - Region where the CloudWatchLogs is locat
  endpoint: # [string] Endpoint - CloudWatchLogs service endpoint. If empty, defa
  signatureVersion: # [string] Signature version - Signature version to use for s
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
  enableAssumeRole: # [boolean] Enable for CloudWatchLogs - Use Assume Role crede
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches befo
  maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or 

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
minio_output: # [object]
  type: # [string] Output Type
  endpoint: # [string] [required] MinIO endpoint - MinIO service url (e.g. http:/
  bucket: # [string] MinIO bucket name - Name of the destination MinIO bucket. Th
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication

  # -------------- 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

  # --------------------------------------------------------

  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 whic
  addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging loc
  removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging directo

  # -------------- if removeEmptyDirs is true ---------------
  #                 if removeEmptyDirs is true

  emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to cle

  # ---------------------------------------------------------

  destPath: # [string] [required] Key prefix - Root directory to prepend to path
  signatureVersion: # [string] Signature version - Signature version to use for s
  objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
  storageClass: # [string] Storage class - Storage class to select for uploaded o
  serverSideEncryption: # [string] Server-side encryption - Server-side encryptio
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
  partitionExpr: # [string] Partitioning expression - JS expression defining how
  format: # [string] Data format - Format of the output data.

  # -------------- if format is json ---------------

  compress: # [string] Compress - Choose data compression format to apply before

  # ---------------------------------------------------------



  # -------------- if format is parquet ---------------

  parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
  parquetRowGroupSize: # [string] Row group size - Ideal memory size for row grou
  parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.
  spacer: # [null]
  parquetVersion: # [string] Parquet version - Determines which data types are su
  parquetDataPageVersion: # [string] Data page version - Serialization format of
  shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique row

  # ---------------------------------------------------------

  baseFileName: # [string] File name prefix expression - JavaScript expression to
  fileNameSuffix: # [string] File name suffix expression - JavaScript expression
  maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
  maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of tim
  maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of tim
  maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
  onBackpressure: # [string] Backpressure behavior - Whether to block or drop eve
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
```

```
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
statsd_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thro
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # --------------------------------------------------------

  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
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to s
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
statsd_ext_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thro
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # --------------------------------------------------------

  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
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to s
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
graphite_output: # [object]
  type: # [string] Output Type
  protocol: # [string] Destination Protocol - Protocol to use when communicating

  # -------------- if protocol is tcp ---------------

  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to thro
  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # --------------------------------------------------------

  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
  flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to s
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
router_output: # [object]
  type: # [string] Output Type
  rules: # [array] Rules - Event routing rules
    - filter: # [string] Filter Expression - JavaScript expression to select ever
      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 b
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
sqs_output: # [object]
  type: # [string] Output Type
  queueName: # [string] Queue Name - The name, URL, or ARN of the SQS queue to se
  queueType: # [string] [required] Queue Type - The queue type used (or created).
  awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Lea
  messageGroupId: # [string] Message Group ID - This parameter applies only to FI
  createQueue: # [boolean] Create Queue - Create queue if it does not exist.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication

  # -------------- if awsAuthenticationMethod is manual ---------------
  # -------------- if awsAuthenticationMethod is manual ---------------

  awsApiKey: # [string] Access key - Access key

  # --------------------------------------------------------

  # -------------- if awsAuthenticationMethod is secret ---------------

  awsSecret: # [string] Secret key pair - Select (or create) a stored secret that

  # --------------------------------------------------------

  awsSecretKey: # [string] Secret key - Secret key
  region: # [string] Region - AWS Region where the SQS queue is located. Required
  endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS
  signatureVersion: # [string] Signature version - Signature version to use for s
  reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
  rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to r
  enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to a
  assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the ro
  assumeRoleExternalId: # [string] External ID - External ID to use when assuming
  maxQueueSize: # [number] Max queue size - Maximum number of queued batches befo
  maxRecordSizeKB: # [number] Max record size (KB) - Maximum size (KB) of batches
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  maxInProgress: # [number] Max concurrent requests - The maximum number of in-pr
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
snmp_output: # [object]
```

```
snmp_output: # [object]
  type: # [string] Output Type
  hosts: # [array] SNMP Trap Destinations - One or more SNMP destinations to for
    - host: # [string] Address - Destination host
      port: # [number] Port - Destination port, default is 162
  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatica
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
sumo_logic_output: # [object]
  type: # [string] Output Type
  url: # [string] API URL - Sumo Logic HTTP collector URL to which events should
  customSource: # [string] Custom source name - Optionally, override the source
  customCategory: # [string] Custom source category - Optionally, override the s
  concurrency: # [number] Request concurrency - Maximum number of ongoing reques
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
  maxPayloadEvents: # [number] Max events per request - Max number of events to
  compress: # [boolean] Compress - Whether to compress the payload body before s
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each qu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the qu
  pqPath: # [string] Queue file path - The location for the persistent queue fil
  pqCompress: # [string] Compression - Codec to use to compress the persisted da
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop ev
  pqControls: # [object]

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatica
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
datadog_output: # [object]
  type: # [string] Output Type
  contentType: # [string] Send logs as - The content type to use when sending log
  message: # [string] Message field - Name of the event field that contains the r
  source: # [string] Source - Name of the source to send with logs. When you send
  host: # [string] Host - Name of the host to send with logs. When you send logs
  service: # [string] Service - Name of the service to send with logs. When you s
  tags: # [array of strings] Datadog tags - List of tags to send with logs (e.g.,
  allowApiKeyFromEvents: # [boolean] Allow API key from events - If enabled, the
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  severity: # [string] Severity - Default value for message severity. When you se
  site: # [string] Datadog site
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 l
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # -------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select
  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

  # -------------- if authType is secret ---------------


  # ---------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
grafana_cloud_output: # [object]
  type: # [string] Output Type
  lokiUrl: # [string] Loki URL - The endpoint to send logs to, e.g.: https://logs
  prometheusUrl: # [string] [required] Prometheus URL - The remote_write endpoint
  message: # [string] Logs message field - Name of the event field that contains
  messageFormat: # [string] Message Format - Which format to use when sending log

  # -------------- if messageFormat is json ---------------

  compress: # [boolean] Compress - Whether to compress the payload body before se

  # ---------------------------------------------------------

  labels: # [array] Logs labels - List of labels to send with logs. Labels define
    - name: # [string] Name - Name of the label.
      value: # [string] Value - Value of the label.
  metricRenameExpr: # [string] Metrics renaming expression - A JS expression that
  prometheusAuth: # [object]
    authType: # [string] Authentication Type - The authentication method to use f

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization h

    # ---------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
    textSecret: # [string] Auth token (text secret) - Select (or create) a storec

    # ---------------------------------------------------------



    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud doma

    # ---------------------------------------------------------



    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secre

    # ---------------------------------------------------------

  lokiAuth: # [object]
    authType: # [string] Authentication Type - The authentication method to use f

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization h

    # ---------------------------------------------------------



    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a storec

    # ---------------------------------------------------------



    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud doma

    # ---------------------------------------------------------
```

```
      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secre

      # -------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing request
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
    maxPayloadEvents: # [number] Max events per request - Maximum number of events
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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 l
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    systemFields: # [array of strings] System fields - Set of fields to automatical
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # -------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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
    messageFormat: # [string] Message Format - Which format to use when sending log

    # -------------- if messageFormat is json ---------------
```

```
compress: # [boolean] Compress - Whether to compress the payload body before se

# ---------------------------------------------------------

labels: # [array] Logs labels - List of labels to send with logs. Labels define
  - name: # [string] Name - Name of the label.
    value: # [string] Value - Value of the label.
authType: # [string] Authentication Type - The authentication method to use fo

# -------------- if authType is token ----------------

token: # [string] Auth token - Bearer token to include in the authorization hea

# ---------------------------------------------------------


# -------------- if authType is textSecret ----------------

textSecret: # [string] Auth token (text secret) - Select (or create) a stored

# ---------------------------------------------------------


# -------------- if authType is basic ----------------

username: # [string] Username - Username for authentication
password: # [string] Password - Password (a.k.a API key in Grafana Cloud domai

# ---------------------------------------------------------


# -------------- if authType is credentialsSecret ----------------

credentialsSecret: # [string] Credentials secret - Select (or create) a secret

# ---------------------------------------------------------

concurrency: # [number] Request concurrency - Maximum number of ongoing reques
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
maxPayloadEvents: # [number] Max events per request - Maximum number of events
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
```

```
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
    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 l
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
    systemFields: # [array of strings] System fields - Set of fields to automatical
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
    pqPath: # [string] Queue file path - The location for the persistent queue file
    pqCompress: # [string] Compression - Codec to use to compress the persisted dat
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  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
    sendMetadata: # [boolean] Send metadata - Whether to generate and send metadata

    # -------------- if sendMetadata is true ---------------

    metricsFlushPeriodSec: # [number] Metadata flush period (sec) - How frequently

    # --------------------------------------------------------

    systemFields: # [array of strings] System fields - Set of fields to automatical
    concurrency: # [number] Request concurrency - Maximum number of ongoing request
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
    maxPayloadEvents: # [number] Max events per request - Max number of events to i
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
```

```
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 l
failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or c

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
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

# -------------------------------------------------------
```

```
    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text s

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    environment: # [string] Environment - Optionally, enable this config only on a
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  ring_output: # [object]
    type: # [string] Output Type
    format: # [string] Data format - Format of the output data.
    partitionExpr: # [string] Partitioning expression - JS expression to define how
    maxDataSize: # [string] Max data size - Maximum disk space allowed to be consum
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data (e
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults to
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop eve
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
    systemFields: # [array of strings] System fields - Set of fields to automatical
    environment: # [string] Environment - Optionally, enable this config only on a
  open_telemetry_output: # [object]
    type: # [string] Output Type
    endpoint: # [string] Endpoint - The endpoint to send OTEL events to. It can be
    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

# --------------------------------------------------------



# -------------- if authType is textSecret ---------------

textSecret: # [string] Token (text secret) - Select (or create) a stored text

# --------------------------------------------------------

tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that ar
  certificateName: # [string] Certificate name - The name of the predefined ce
  caPath: # [string] CA certificate path - Path on client in which to find CA
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in w
  certPath: # [string] Certificate path (mutual auth) - Path on client in whic
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
  minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

  # --------------------------------------------------------

metadata: # [array] Metadata - Extra information to send with each gRPC reques
  - key: # [string] Key - The key of the metadata.
    value: # [string] Value - The value of the metadata.
concurrency: # [number] Request concurrency - Maximum number of ongoing reques
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
failedRequestLoggingMode: # [string] Failed request logging mode - Determines
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
connectionTimeout: # [number] Connection Timeout - Amount of time (millisecond
keepAliveTime: # [number] Keep Alive Time (seconds) - How often the sender sho
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

# -------------- if onBackpressure is queue ---------------
```

```
  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
dataset_output: # [object]
  type: # [string] Output Type
  messageField: # [string] Message field - Name of the event field that contains
  excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
  serverHostField: # [string] Server/host field - Name of the event field that co
  timestampField: # [string] Timestamp field - Name of the event field that conta
  defaultSeverity: # [string] Severity - Default value for event severity. If the
  site: # [string] DataSet site - DataSet site to which events should be sent
  customUrl: # [string]
  concurrency: # [number] Request concurrency - Maximum number of ongoing request
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
  maxPayloadEvents: # [number] Max events per request - Max number of events to i
  compress: # [boolean] Compress - Whether to compress the payload body before se
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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 l
  failedRequestLoggingMode: # [string] Failed request logging mode - Determines w
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ----------------------------------------------------------

  authType: # [string] Authentication method - Enter API key directly, or select
  apiKey: # [string] API key - A 'Log Write Access' API key for the DataSet accou

  # -------------- if authType is manual ---------------



  # ----------------------------------------------------------

  textSecret: # [string] API key (text secret) - Select (or create) a stored text

  # -------------- if authType is secret ---------------



  # ----------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
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 curr
  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
      servername: # [string] TLS Servername - Servername to use if establishing a
      weight: # [number] Load Weight - The weight to use for load-balancing purpo
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
  maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent

  # ----------------------------------------------------------

  compression: # [string] Compression - Codec to use to compress the data before
  throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throt
  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
    servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
    certificateName: # [string] Certificate name - The name of the predefined cer
    caPath: # [string] CA certificate path - Path on client in which to find CA c
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
    certPath: # [string] Certificate path (mutual auth) - Path on client in which
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # --------------------------------------------------------

  connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
  writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
  tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befo
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
  pqPath: # [string] Queue file path - The location for the persistent queue file
  pqCompress: # [string] Compression - Codec to use to compress the persisted dat
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
  pqControls: # [object]

  # ----------------------------------------------------------
```

```
  authType: # [string] Authentication method - Enter a token directly, or provide
  authToken: # [string] Auth token - Optional authentication token to include as

  # -------------- if authType is manual ---------------


  # --------------------------------------------------------

  textSecret: # [string] Auth token (text secret) - Select (or create) a stored t

  # -------------- if authType is secret ---------------


  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatical
  environment: # [string] Environment - Optionally, enable this config only on a
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
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 curr
  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
      servername: # [string] TLS Servername - Servername to use if establishing a
      weight: # [number] Load Weight - The weight to use for load-balancing purpo
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve an
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
  maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
```

```
# ----------------------------------------------------------

compression: # [string] Compression - Codec to use to compress the data before
throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throt
tls: # [object] TLS settings (client side)
  disabled: # [boolean] Disabled

  # -------------- if disabled is false ---------------

  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  servername: # [string] Server name (SNI) - Server name for the SNI (Server Na
  certificateName: # [string] Certificate name - The name of the predefined cer
  caPath: # [string] CA certificate path - Path on client in which to find CA c
  privKeyPath: # [string] Private key path (mutual auth) - Path on client in wh
  certPath: # [string] Certificate path (mutual auth) - Path on client in which
  passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
  minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
  maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

  # ----------------------------------------------------------

connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds
writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes befo
excludeFields: # [array of strings] Exclude Fields - Fields to exclude from the
onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
pqControls: # [object]

# ----------------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data before sending out to
systemFields: # [array of strings] System fields - Set of fields to automatical
environment: # [string] Environment - Optionally, enable this config only on a
streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
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.
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS

  # --------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the cur
  urls: # [array] Cribl Worker Endpoints
    - url: # [string] Cribl Endpoint - URL of a Cribl Worker to send events to,
      weight: # [number] Load Weight - The weight to use for load-balancing purp
  dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve a
  loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - Ho

  # --------------------------------------------------------

  tls: # [object] TLS settings (client side)
    disabled: # [boolean] Disabled

    # -------------- if disabled is false ---------------

    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that ar
    servername: # [string] Server name (SNI) - Server name for the SNI (Server N
    certificateName: # [string] Certificate name - The name of the predefined ce
    caPath: # [string] CA certificate path - Path on client in which to find CA
    privKeyPath: # [string] Private key path (mutual auth) - Path on client in w
    certPath: # [string] Certificate path (mutual auth) - Path on client in whic
    passphrase: # [string] Passphrase - Passphrase to use to decrypt private key
    minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
    maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when

    # --------------------------------------------------------

  tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes bef
  excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
  compression: # [string] Compression - Codec to use to compress the data before
  concurrency: # [number] Request concurrency - Maximum number of ongoing reques
  maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
  maxPayloadEvents: # [number] Max events per request - Max number of events to
  rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
  timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait f
  flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
  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
  safeHeaders: # [array of strings] Safe headers - List of headers that are safe
  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
  onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or

  # -------------- if onBackpressure is queue ---------------

  pqMaxFileSize: # [string] Max file size - The maximum size to store in each qu
  pqMaxSize: # [string] Max queue size - The maximum amount of disk space the qu
  pqPath: # [string] Queue file path - The location for the persistent queue fil
  pqCompress: # [string] Compression - Codec to use to compress the persisted da
  pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop ev
  pqControls: # [object]

  # --------------------------------------------------------

  pipeline: # [string] Pipeline - Pipeline to process data before sending out to
  systemFields: # [array of strings] System fields - Set of fields to automatica
  environment: # [string] Environment - Optionally, enable this config only on a
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 event
  useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS

  # --------------------------------------------------------



  # -------------- if loadBalanced is true ---------------

  excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the cur
```

```
urls: # [array] Humio HEC Endpoints
  - url: # [string] HEC Endpoint - URL to a Humio HEC endpoint to send events t
    weight: # [number] Load Weight - The weight to use for load-balancing purpo
dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve ar
loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How

# ----------------------------------------------------------

concurrency: # [number] Request concurrency - Maximum number of ongoing request
maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the r
maxPayloadEvents: # [number] Max events per request - Max number of events to i
compress: # [boolean] Compress - Whether to compress the payload body before se
rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait fo
flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
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 w
safeHeaders: # [array of strings] Safe headers - List of headers that are safe
format: # [string] Request Format - Send data in JSON format to the api/v1/inge
authType: # [string] Authentication method - Enter a token directly, or provide

# -------------- if authType is manual ---------------

token: # [string] HEC Auth token - Humio HEC authentication token

# ----------------------------------------------------------



# -------------- if authType is secret ---------------

textSecret: # [string] HEC Auth token (text secret) - Select (or create) a stor

# ----------------------------------------------------------

onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or q

# -------------- if onBackpressure is queue ---------------

pqMaxFileSize: # [string] Max file size - The maximum size to store in each que
pqMaxSize: # [string] Max queue size - The maximum amount of disk space the que
pqPath: # [string] Queue file path - The location for the persistent queue file
```

```
[illegible]
pqCompress: # [string] Compression - Codec to use to compress the persisted dat
pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop eve
pqControls: # [object]

# ----------------------------------------------------------

pipeline: # [string] Pipeline - Pipeline to process data before sending out to
systemFields: # [array of strings] System fields - Set of fields to automatical
environment: # [string] Environment - Optionally, enable this config only on a
streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

19.6.15. 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, i
```

19.6.16. Perms.Yml

`perms.yml` stores Permissions configured for Cribl Stream users.

```
perms:
  - gid:    # [string]
    id:     # [string] Not used if type is 'groups'.
    type:   # [string] Resource type, one of: 'groups' | 'datasets' | 'dataset-prov
    policy: # [string] Policy name.
```

19.6.17. 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
```

19.6.18. 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.
```

19.6.19. 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 everythi
  policy:
    - '* *'
reader_all:
  description: 'Members with reader_all role get read-only access to all Worker Gro
  policy:
    - GroupRead *
collect_all:
  description: 'Members of this group can run existing collection jobs of all Worke
  policy:
    - GroupCollect *
editor_all:
  description: 'Members with editor_all role get read/write access to all Worker Gr
  policy:
    - GroupEdit *
owner_all:
  description: 'Members with owner_all role get read/write access as well as Deploy
  policy:
    - GroupFull *
user:
  description: 'The base user role allows users to see the system info along with t
  policy:
    - GET /system/info
    - GET /system/info/*
    - GET /system/users
    - GET /system/instance/distributed
    - GET /system/instance/distributed/*
    - GET /clui
    - PATCH /ui/*
```

19.6.20. 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 w
  description: # [string] Description - Brief description of this sample file. Opti
  ttl: # [number] Expiration (hours) - Time to live for the sample, the TTL is rese
  tags: # [string] Tags - One or more tags related to this sample file. Optional.
```

The corresponding sample files reside in `$CRIBL_HOME/data/samples`.

19.6.21. 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-
```

19.6.22. 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 scr
  env: # [object] Env Variables - Extra environment variables to set when executin
```

19.6.23. Secrets.Yml

`secrets.yml` stores secrets for Cribl Stream.

The secrets are decrypted and encrypted using a `cribl.secret` file. `cribl.secret` is unique per Worker Group and resides in the `$CRIBL_HOME/groups/<group-name>/local/cribl/auth` directory on Leader nodes, or in `$CRIBL_HOME/local/cribl/auth` for Workers or single-instance deployments.

```
<secret-id>:
  secretType: # [string] One of: keypair | text | credentials
  secrets:
    # -------------- if secretType is keypair ----------------
    apiKey: "<hashed-value>"
    secretKey: "<hashed-value>"

    # -------------- if secretType is text -------------------
    value: "<hashed-value>"

    # -------------- if secretType is credentials -----------
    username: "<hashed-value>"
    password: "<hashed-value>"
```

19.6.24. Service.Yml

`service.yml` maintains configuration for Cribl Stream service processes.

In the UI, you can configure them on the Leader at **Settings** > **Global Settings** > **System** > **Service Processes** > **Services**.

Be careful about reducing the predefined limits. Extremely low values can prevent some Cribl Stream components from functioning.

$CRIBL_HOME/default/cribl/service.yml

```
connections: # [Object] - Heap memory limit for the connection listener processes.
  procs: # [number; default: 1, minimum: 1]
  memoryLimit: # [string; default: 2GB]
metrics: # [Object] - Heap memory limit for the metrics process
  procs: # [number; default: 1] — Single process; values other than 1 will be treat
  memoryLimit: # [string; default: 2GB]
notifications: # [Object] - Heap memory limit for the notifications process
  procs: # [number; default: 1] — Single process; values other than 1 will be treat
  memoryLimit: # [string; default: 2GB]
```

19.6.25. 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. Option
  type: # [string] Type - Type of variable.
  value: # [string] Value - Value of variable
  tags: # [string] Tags - One or more tags related to this variable. Optional.
```

20. Better Practices / Usage Examples

20.1. Tips And Tricks

In designing, supporting, and troubleshooting, complex Cribl Stream deployments and workflows, we've found the following general principles helpful.

Every use case is different. Don't hesitate to contact Cribl Support for help with solving your specific needs.

Architecture and Deployment

Separate Data by Worker Groups, Routes, or Both?

Separate your data volume by Worker Groups, or by Routes, or both. Common business reasons for sorting by Groups include data isolation by business group, ensuring data privacy, and achieving cost savings through geo-isolation. Another consideration is: Which type of separation will be the easiest to understand and manage as your organization grows?

Connecting Cribl Stream Groups/Instances with Same Leader:

Cribl HTTP or Cribl TCP

Where Workers share the same Leader, the obvious choices for sending data between those Workers are the Cribl HTTP Source and Destination pair, or the Cribl TCP Source and Destination pair. Either option prevents double-counting of this internal data flow against your on-prem license or Cribl.Cloud plan.

Connecting Stream Groups/Instances with Different Leaders:

TCP JSON

When sending between Cribl Stream Worker Groups and/or instances connected to **different** Leaders, use the TCP JSON Source and Destination pair. While Cribl Stream supports multiple Sources/Destinations for sending and receiving, TCP JSON is ideal because it supports TLS, has solid compression, and is overall well-formatted to maintain data's structure between sender and receiver.

Input Side: Event Breakers and Timestamping

Validate timestamping and event breaking before turning on a Source.

If a Source supports Event Breakers (e.g., AWS Sources), it is much more efficient to perform JSON unroll in a Breaker, versus in a Pipeline Function.

Routes

Avoid Route Creep

Design data paths to move through as few Routes as possible. This usually means we want to reduce volume of events as early as possible. If most of the events being processed are sent to a Destination by the first Route, this will spare all the other Routes and Pipelines wasted processing cycles and testing against whether filter criteria are met.

Prevent Function Creep

As a mirror principle, clone events as late as possible in the Routes. This will minimize the number of Functions acting on data, to the extent possible.

Leave No Data Behind

Create a catchall Route at the end of the list to explicitly route events that fail to match any Route filters.

Pipelines and Functions Logic

Don't Overload Pipelines

Do not use the same Pipeline for both pre-processing and post-processing. This makes isolation and troubleshooting extremely difficult.

Extract or Parse by Desired Yield

If you need to extract one or just a few fields, use the Regex Extract Function. If you need to extract all or most of an event's fields, use the Parser Function.

Use Function Groups for Legibility

Function groups might be helpful in organizing your Pipeline. These groups are abstractions, purely for visual context, and do not affect the movement of data through Functions. Data will move down the listed Functions, ignoring any grouping assignments.

Use Comments to Preserve Legibility

Comment, comment, comment. There is a lot of contextual information which might become lost over time as users continue to advance and add Routes and Pipelines to Cribl Stream. A good principle is to keep the design decisions as simple and easy to understand as possible, and to document the assumptions around each Route and Pipeline in comments as clearly as possible.

Create Expressions Around Fields with Non-Alphanumeric Characters

If there are fields with 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.) For more details, see MDN's Property Accessors documentation. In any other place where such fields are referenced – e.g., in an Eval Function's Field names – you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Specify Fields' Precedence Order in Expressions

In any Source that supports adding **Fields (Metadata)**, your **Value** expression can specify that fields in events should override these fields' values. E.g., the following expression's L->R/OR logic specifies that if an inbound event includes an `index` field, use that field's value; otherwise, fall back to the `myIndex` constant defined here: `` `${__e['index'] || 'myIndex'}` `` .

Break Up `_raw`

Consider avoiding the use of `_raw` as a temporary location for data. Instead, split out explicitly separate fields/variables.

Optimize PQ Using File Size

Consider using a smaller maximum file size in Persistent Queues settings, for better buffering.

Output Side

Although this might be obvious: Ship metrics out to a dedicated alerting/metrics engine (ELK, Grafana, Splunk, etc.)

Troubleshooting

Troubleshoot streams processing systems from right to left. Start at the Destination, and check for block status from the Destination back to the Source.

Don’t run health checks on data ports too frequently, as this can lead to false-positives errors.

20.2. Lookup Applications

20.2.1. Ingest-Time Lookups

Enriching Data in Motion

To enrich events with new fields from external sources (such as `.csv` files), we use Cribl Stream's out-of-the-box Lookup Function. Ingestion-time lookups are not only great for normalizing field names and values, but also ideal for use cases where:

- Fast access via the looked-up value is required. For example, when you don't have a `datacenter` field in your events, but you do have a `host-to-datacenter` map, and you need to search by `datacenter`.
- Looked-up information must be temporally correct. For example, assume that you have a highly dynamic infrastructure, and you need to resolve a resource name (e.g., a container name) to its address. You can't afford to defer this to search time/runtime, as the resource and its records might no longer exist.

To use large binary databases (like GeoIP `.mmdb` files) for Cribl Stream lookups, see Managing Large Lookups. To achieve faster lookups, use Cribl Stream's Redis Function.

Working with Lookups – Example 1

Let's assume we have the following lookup file. Given the field `conn_state` in an event, we would like to add a corresponding ingestion-time field called `action`.

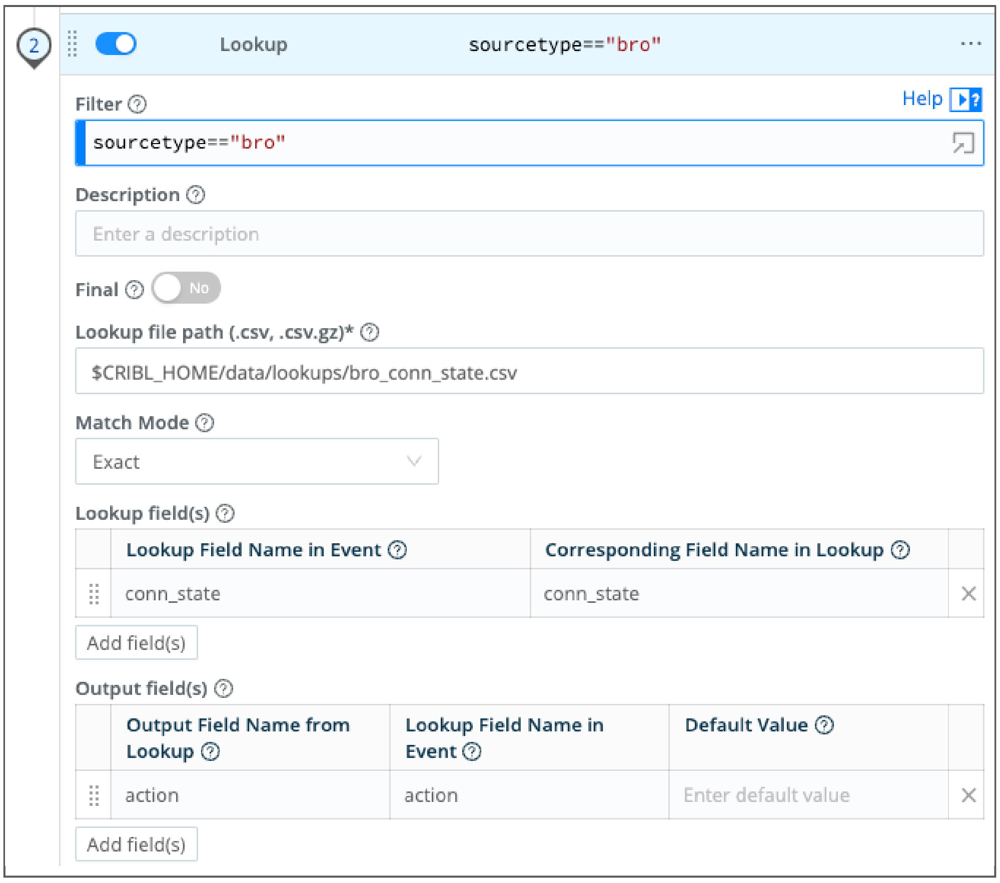
bro_conn_state.csv

```
action,"conn_state","conn_state_meaning"
dropped,S0,"Connection attempt seen, no reply."
allowed,S1,"Connection established, not terminated."
allowed,SF,"Normal establishment and termination."
blocked,REJ,"Connection attempt rejected."
allowed,S2,"Connection established and close attempt by originator seen (but no rep
allowed,S3,"Connection established and close attempt by responder seen (but no repl
allowed,RSTO,"Connection established, originator aborted (sent a RST)."
allowed,RSTR,"Established, responder aborted."
dropped,RSTOS0,"Originator sent a SYN followed by a RST, we never saw a SYN-ACK fro
dropped,RSTRH,"Responder sent a SYN ACK followed by a RST, we never saw a SYN from
dropped,SH,"Originator sent a SYN followed by a FIN, we never saw a SYN ACK from th
dropped,SHR,"Responder sent a SYN ACK followed by a FIN, we never saw a SYN from th
allowed,OTH,"No SYN seen, just midstream traffic (a 'partial connection' that was n
```

First, make sure you have a Route and Pipeline configured to match desired events.

Next, let's add a **Lookup** function to the Pipeline, with these settings:

- **Lookup file path**: `$CRIBL_HOME/data/lookups/bro_conn_state.csv` (note that Environment variables are allowed in the path).
- **Lookup Field Name in Event** set to `conn_state`.
- **Corresponding Field Name in Lookup** set to `conn_state`.
- **Output Field Name from Lookup** set to `action`.
- **Lookup Field Name in Event** set to `action`.


2 Lookup sourcetype=="bro"

Filter: sourcetype=="bro"

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: $CRIBL_HOME/data/lookups/bro_conn_state.csv

Match Mode: Exact

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
conn_state	conn_state

Add field(s)

Output field(s)

Output Field Name from Lookup	Lookup Field Name in Event	Default Value
action	action	Enter default value

Add field(s)

Lookup Function to add `action` field

To confirm success, verify that this search returns expected results: `sourcetype="bro" action::allowed`. Change the `action` value as necessary.

Working with Lookups – Example 2

Let's assume we have the following lookup file, and given **both** the fields `impact` and `priority` in an event, we would like to add a corresponding ingestion-time field called `severity`.

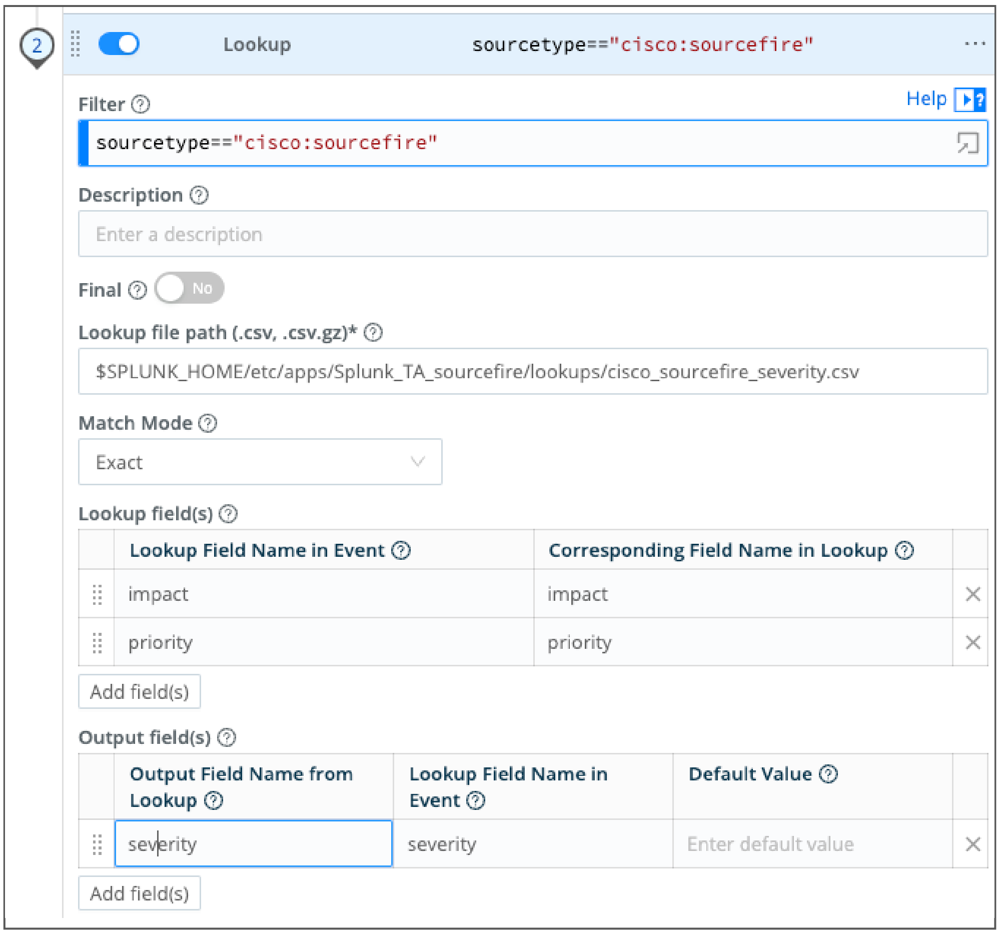
cisco_sourcefire_severity.csv

```
impact,priority,severity
1,high,critical
2,high,critical
3,high,high
4,high,high
0,high,high
"*",high,high
.....
"*",medium,medium
1,low,medium
2,low,medium
3,low,low
4,low,low
0,low,low
"*",low,low
1,none,low
2,none,low
3,none,informational
4,none,informational
0,none,informational
"*",none,informational
```

First, make sure you have a Route and Pipeline configured to match desired events.

Next, let's add a **Lookup** function to the Pipeline, with these settings:

- **Lookup file path**: `$SPLUNK_HOME/etc/apps/Splunk_TA_sourcefire/lookups/cisco_sourcefire_severity.csv` (note that Environment variables are allowed in the path).
- **Lookup Field Name(s) in Event** set to `impact` and `priority`.
- **Corresponding Field Name(s) in Lookup** set to `impact` and `priority`.
- **Output Field Name from Lookup** set to `severity`.
- **Lookup Field Name in Event** set to `severity`.



Lookup `sourcetype=="cisco:sourcefire"`

Filter: `sourcetype=="cisco:sourcefire"`

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: $SPLUNK_HOME/etc/apps/Splunk_TA_sourcefire/lookups/cisco_sourcefire_severity.csv

Match Mode: Exact

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
impact	impact
priority	priority

Output field(s)

Output Field Name from Lookup	Lookup Field Name in Event	Default Value
severity	severity	Enter default value

Lookup Function to add `severity` field

To confirm success, verify that this search returns expected results: `sourcetype="cisco:sourcefire" severity::medium`. Change the `severity` value as necessary.

20.2.2. Managing Large Lookups

This page offers a general approach to managing large lookup files. While Cribl Stream's Git integration normally helps manage configuration changes, large lookups are exceptions. In many cases, you might want to exclude these files from Git, to reduce excessive deploy traffic. This approach can also prevent Git Push commands from encountering large file errors.

Good scenarios for this approach are:

- Large binary files – like databases – which don't benefit from Git's typical efficient storage of only the deltas between versions. (With binary files, Git must replace the whole file for each new version.)
- Files updated frequently and/or files updated independent of Cribl Stream.
- Files replicated on many Worker Nodes.

The steps below assume access to a command line and (more importantly) to your OS' filesystem. Where you lack such access – for example, in a Cribl.Cloud deployment – load lookup files of all sizes via Cribl Stream's UI, as outlined in Lookups Library.

About the MaxMind GeoLite Example

We'll illustrate this with an example that often combines all three conditions: setting up the free, popular MaxMind GeoLite2 City database to support Cribl Stream's GeoIP lookup Function. This example anticipates a Cribl Stream production distributed deployment, where the GeoLite database is updated nightly across multiple Workers.

This example includes complete instructions for this particular setup. However, you can generalize the example to other MaxMind databases, and to other large lookup files – including large `.csv`'s that similarly receive frequent updates.

Reducing Deploy Traffic

The general approach for handling large lookups is:

- Do not place these files in the standard `$CRIBL_HOME/data/lookups`.
- Instead, place them in a `$CRIBL_HOME` subdirectory that's excluded from Git version control, through inclusion in the `$CRIBL_HOME/.gitignore` file. Deploying the files to the Leader Node and all desired Workers will require a manual procedure and will be required for the initial deployment as well as subsequent updates.

The example below uses `$CRIBL_HOME/state` subdirectory, which is already listed in the default `.gitignore` file that ships with Cribl Stream.

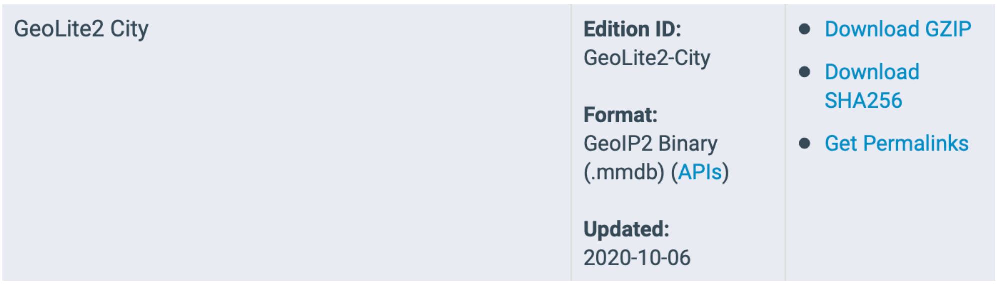
> If you prefer, you can use a different path, which might be a path outside `$CRIBL_HOME`. If you choose this alternative, be sure to add that path to `.gitignore`.
>
> However, Cribl recommends using a `$CRIBL_HOME` subdirectory like `$CRIBL_HOME/state`, because this inherits appropriate permissions and simplifies backup/restore operations.

Let's move on to the MaxMind GeoLite specifics.

Download and Extract the Database

To enable the GeoIP Function using the MaxMind GeoLite 2 City database, your first steps are:

1. Create a free MaxMind account, at the page linked above.
2. Log in to your MaxMind account portal and select **Download Databases**.
3. On the Download page, look for the database you want. (In this example, you'd locate the **GeoLite2 City** section.) Note the **Format: GeoIP2 Binary**, and select **Download GZIP**.



GeoLite2 City	**Edition ID:** GeoLite2-City **Format:** GeoIP2 Binary (.mmdb) (APIs) **Updated:** 2020-10-06	● Download GZIP ● Download SHA256 ● Get Permalinks

GeoLite2 City database: Download binary GZIP

4. Extract the archive to your local system.
5. Change to the directory created when you extracted the archive. This directory's name will correspond to the date you downloaded the file, so in the above `2020-10-06` example, you would use:
   `$ cd GeoLite2-City_20201006`

Copy the Database File to the Leader and Worker Nodes (Recommended)

In distributed deployments, Cribl recommends copying the MaxMind database separately to the Leader and all Worker Nodes. For example, placing it in the `$CRIBL_HOME/state` path minimizes the Git commit/deploy overhead for nightly updates to the binary database file.

Once in the database's directory, execute commands of this form:

```
$ scp *.mmdb <user>@<leader-node>:
$ scp *.mmdb <user>@<worker-node>:
```

Copy the file to each Worker in the Worker Group where you intend to use Cribl Stream's GeoIP Function.

The above commands copy the `.mmdb` database file into your user's home directory on each Node. Next, we'll move it to `$CRIBL_HOME/state` on each Node. Execute these commands on both the Leader and Worker Nodes:

```
$ sudo mv ~/*.mmdb <$CRIBL_HOME>/state/
$ sudo chown -R cribl:cribl <$CRIBL_HOME>/state/
```

Now that the database is in place, your Pipelines can use the GeoIP Function to enrich data. In the Function's **GeoIP file (.mmdb)** field, insert the complete `$CRIBL_HOME/state/<filename>.mmdb` file path.

Copy the Database File Only to the Leader (Alternative)

In smaller deployments, you might choose to copy this MaxMind database only to the Leader Node, and to let Workers receive updates via Git commit/deploy. In this case, the final commands above might look like this:

Shell

```
$ sudo cp ~/*.mmdb /opt/cribl/groups/<group-name>/data/lookups/
$ cd /opt/cribl/groups/<group-name>/data/lookups/
$ sudo chown cribl:cribl *.mmdb
```

Automatic Updates to the MaxMind Database

Use the GeoIP Update program MaxMind provides to automatically update your databases. See MaxMind's Automatic Updates for GeoIP2 and GeoIP Legacy Databases documentation for details.

You'll need two modifications specific to Cribl Stream:

- GeoIP Update must be set up on the Leader, and on each Worker in any Group that uses GeoIP lookups.
- You need to configure GeoIP Update to point to the directory where the MaxMind database file resides. The program uses a configuration file named `GeoIP.conf` and you utilize the `DatabaseDirectory` parameter to define the location for storing the database file. See MaxMind's GeoIP.conf documentation for details.
  For example, the default setting in `GeoIP.conf` writes output to `/usr/local/share/GeoIP`. You must change this setting to the path where your databases actually reside. If you're using the recommended architecture above, you'd set: `DatabaseDirectory <$CRIBL_HOME>/state/`.

Memory Considerations

Storage aside, large lookup files can also require additional RAM on each Worker Node that processes the lookups. For details, see Memory Sizing for Large Lookups.

20.2.3. Lookups As Filters For Masks

You can make your data architecture more maintainable by using Lookups to route and transform events within Cribl Stream. This use case demonstrates an unusual solution, but one that served one Cribl customer's particular goals (which might overlap with yours):

- Ingest many – hundreds of – different `sourcetype/index` field combinations.
- Send all this data through a common Pipeline.
- Stack four Mask Functions in the Pipeline.
- Evaluate and process each `sourcetype/index` field combination **only** within its applicable Mask Functions – either two or three Masks per combination.

This last restriction reduces latency, by preventing Mask Functions from evaluating non-applicable events, simply to ignore them.

Just to reiterate, this use case outlined here responded to this customer's requirements – one Pipeline combining multiple Mask Functions, for many `sourcetype/index` combinations. More typically, you'd use multiple Pipelines to process different `sourcetype/index` combinations.

To enable this approach, the example below centralizes masking logic for multiple conditions in a Lookup table and corresponding Lookup Functions. The Lookup's output filters events to the applicable Mask Functions. Specifically, we'll show how to instruct Cribl Stream to:

- Check for a particular `index/sourcetype` combination in each event, and
- Based on that combination, determine which Masks to apply to that event.

Design the Lookup

To use a lookup as a filter, you'd start by creating a comma-separated lookup table in this format, and adding it to Cribl Stream:

index_tracker.csv

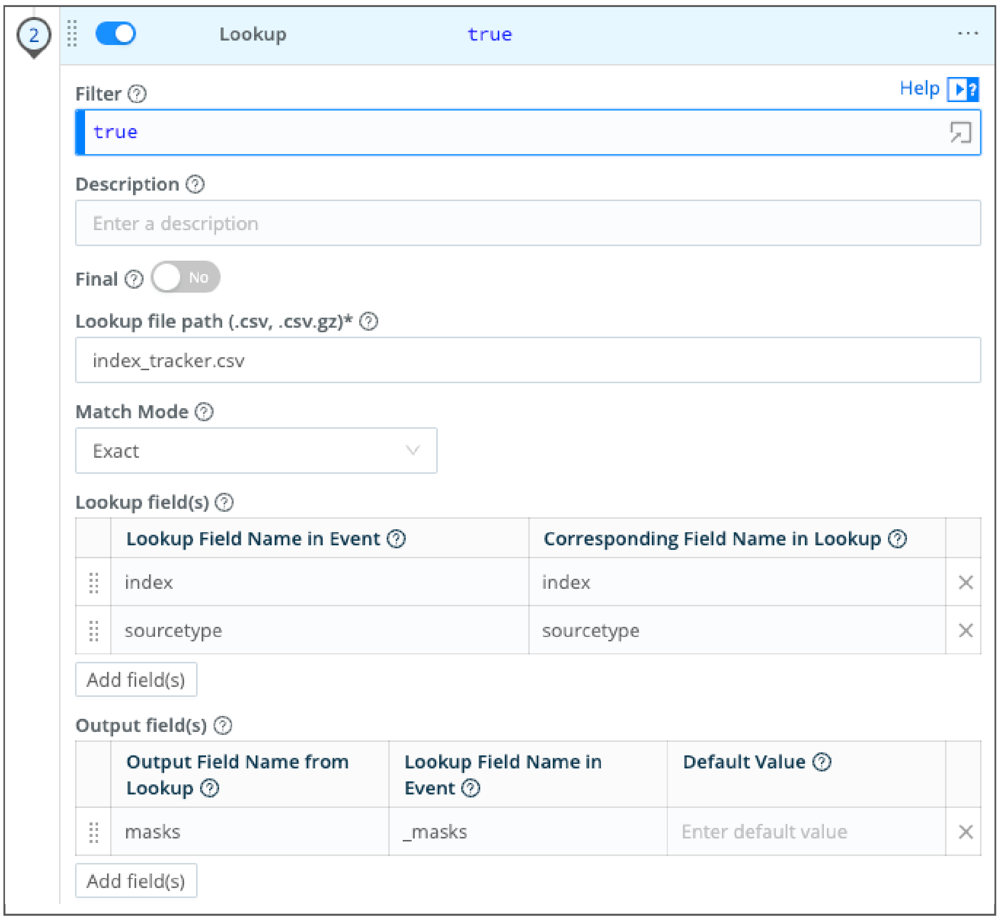
```
index,sourcetype,masks
apache_common, sourcetypec, ssn|credit_card|auth_token
syslog,sourcetypeb,ssn|auth_token
weblog,sourcetypea,auth_token|bearer_token
```

Below the header, each row specifies an index, a sourcetype, and (in the third column) a pipe-delimited list of applicable masks.

To make this example work, the table must have only **one** row for each index/sourcetype combination. (This unusual restriction is particular to this scenario.) So, as you build out the lookup table, you cannot add new masks for **existing** index/sourcetype combinations by appending new rows. Instead, you must modify the third column of the existing rows.

Configure the Pipeline

Create a Cribl Stream Pipeline with a Lookup Function configured like this, pointing to your lookup table:



Filter: true

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: index_tracker.csv

Match Mode: Exact

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
index	index
sourcetype	sourcetype

Output field(s)

Output Field Name from Lookup	Lookup Field Name in Event	Default Value
masks	_masks	Enter default value

Lookup Function's configuration

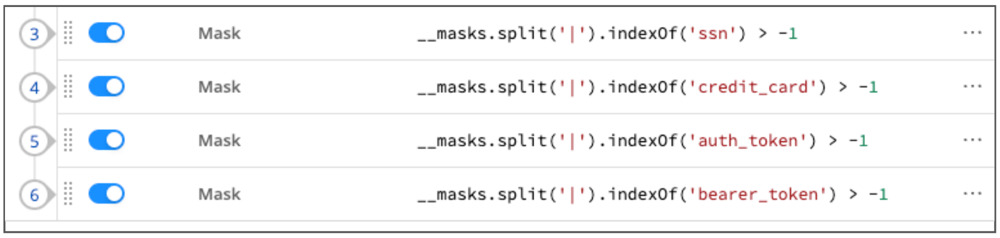
This Function keys against both the `index` and `sourcetype` fields. When it finds a matching combination, it adds a new key-value pair to your event for future filtering.

The key of that key-value pair (namely, `__masks`) starts with a double underscore, to make it a Cribl Stream internal field. This convention ensures that the key-value pair will **not** get passed along to the Destination.

However, you might prefer to export the key-value pair. For example, you might want a Splunk Destination to index the list of masks applied to a given event, alongside that event. (This approach applies to many forensic use cases.) If so, remove the double underscore from the above Function's **Lookup Field Name in Event** value, and from the subsequent Filter expressions for each Mask Function.

Each Mask Function has a JavasScript Filter that breaks the pipe-delimited string into an array, and determines whether the tag for that type of mask (e.g., `bearer_auth`) is in the `__masks` key-value pair. If so, it applies the mask processing. If not, the event moves on to the Pipeline's next Mask Function.

Here are the four Mask Functions below the Lookup Function:



3	Mask	`__masks.split('|').indexOf('ssn') > -1`
4	Mask	`__masks.split('|').indexOf('credit_card') > -1`
5	Mask	`__masks.split('|').indexOf('auth_token') > -1`
6	Mask	`__masks.split('|').indexOf('bearer_token') > -1`

Mask Functions

In this particular example, the pipe-delimited mask tags in the lookup table's third column match the Mask Functions' names, as well as matching their **Filter** conditions. This is just for simplicity – the Functions could have any names, as long as the **Filter** expressions match the tags.

20.2.4. Lookups And Regex Magic

Regular expressions are not just for field extractions – they can also be used inside lookup tables, and in Functions, to replace and manipulate values within fields. Let's walk through a Pipeline that demonstrates four different ways to leverage regular expressions in Cribl Stream.

Why Lookup Tables Matter

When organizations use host naming standards, it is easy to understand things like regions, availability zones (AZs), IP addresses, and more. For example, consider an Amazon host called:

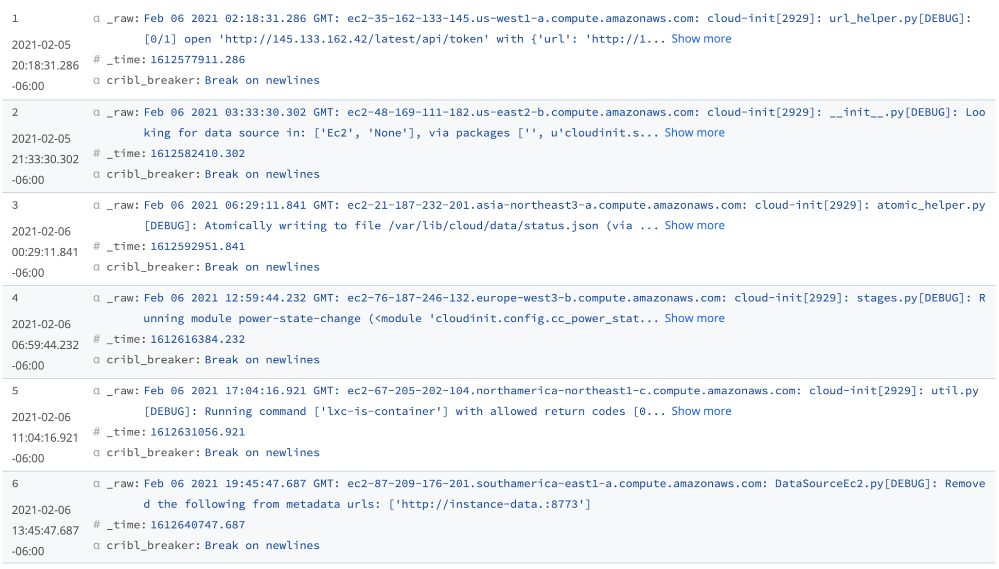
```
ec2-35-162-133-145.us-west1-a.compute.amazonaws.com
```

This is an EC2 host with a (dashed) IP address `35-162-133-145`, in the `us-west1` region, in Availability Zone `a`. You can also see the domain: `compute.amazonaws.com`.

While we can understand the enriched host names, we don't know which indexes to route the data to, nor which sourcetypes to assign to the events, without looking up this information from another source. Doing so is often a huge challenge for organizations. To solve this challenge, let's combine Regex Extract, Lookup, and Eval Functions with some sample events to demonstrate the power of Cribl Stream.

Sample Events

The events below have timestamps broken out, but no indexes, sourcetypes, or other details have been assigned yet:



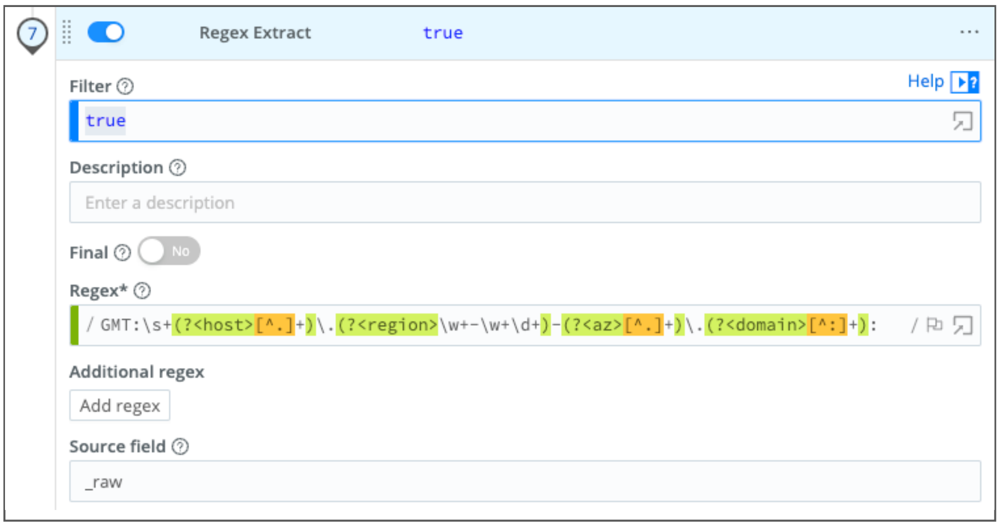
```
1              α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
2021-02-05             [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
20:18:31.286   # _time: 1612577911.286
-06:00         α cribl_breaker: Break on newlines

2              α _raw: Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com: cloud-init[2929]: __init__.py[DEBUG]: Loo
2021-02-05             king for data source in: ['Ec2', 'None'], via packages ['', u'cloudinit.s... Show more
21:33:30.302   # _time: 1612582410.302
-06:00         α cribl_breaker: Break on newlines

3              α _raw: Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaws.com: cloud-init[2929]: atomic_helper.py
2021-02-06             [DEBUG]: Atomically writing to file /var/lib/cloud/data/status.json (via ... Show more
00:29:11.841   # _time: 1612592951.841
-06:00         α cribl_breaker: Break on newlines

4              α _raw: Feb 06 2021 12:59:44.232 GMT: ec2-76-187-246-132.europe-west3-b.compute.amazonaws.com: cloud-init[2929]: stages.py[DEBUG]: R
2021-02-06             unning module power-state-change (<module 'cloudinit.config.cc_power_stat... Show more
06:59:44.232   # _time: 1612616384.232
-06:00         α cribl_breaker: Break on newlines

5              α _raw: Feb 06 2021 17:04:16.921 GMT: ec2-67-205-202-104.northamerica-northeast1-c.compute.amazonaws.com: cloud-init[2929]: util.py
2021-02-06             [DEBUG]: Running command ['lxc-is-container'] with allowed return codes [0... Show more
11:04:16.921   # _time: 1612631056.921
-06:00         α cribl_breaker: Break on newlines

6              α _raw: Feb 06 2021 19:45:47.687 GMT: ec2-87-209-176-201.southamerica-east1-a.compute.amazonaws.com: DataSourceEc2.py[DEBUG]: Remove
2021-02-06             d the following from metadata urls: ['http://instance-data.:8773']
13:45:47.687   # _time: 1612640747.687
-06:00         α cribl_breaker: Break on newlines
```

The Regex Extract Function

Before we can assign an index or sourcetype, we need to extract the `host`, `region`, `az`, and `domain` fields from the events. We can use a Regex Extract Function with this regular expression to extract all four fields:

```
GMT:\s+(?<host>[^.]+)\.(?<region>\w+-\w+\d+)-(?<az>[^.]+)\.(?<domain>[^:]+):
```

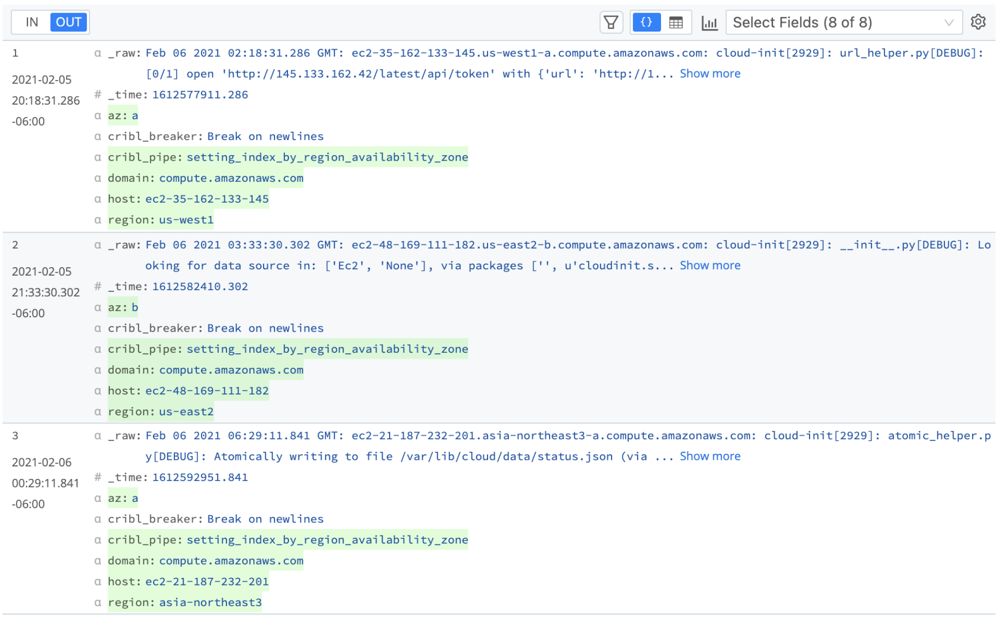
Here's that Regex Extract in a Cribl Stream Pipeline:



```
/ GMT:\s+(?<host>[^.]+)\.(?<region>\w+-\w+\d+)-(?<az>[^.]+)\.(?<domain>[^:]+):  /
```

Results of the Regex Extract Function

On the **OUT** tab of Cribl Stream's Preview pane, the extracted fields of `az`, `domain`, `host`, and `region` now appear below the `_raw` event. You can use these extracted fields for searching in your preferred search solution.



```
1             α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
                    [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
2021-02-05    # _time: 1612577911.286
20:18:31.286  α az: a
-06:00        α cribl_breaker: Break on newlines
              α cribl_pipe: setting_index_by_region_availability_zone
              α domain: compute.amazonaws.com
              α host: ec2-35-162-133-145
              α region: us-west1
2             α _raw: Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com: cloud-init[2929]: __init__.py[DEBUG]: Lo
                    oking for data source in: ['Ec2', 'None'], via packages ['', u'cloudinit.s... Show more
2021-02-05    # _time: 1612582410.302
21:33:30.302  α az: b
-06:00        α cribl_breaker: Break on newlines
              α cribl_pipe: setting_index_by_region_availability_zone
              α domain: compute.amazonaws.com
              α host: ec2-48-169-111-182
              α region: us-east2
3             α _raw: Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaws.com: cloud-init[2929]: atomic_helper.p
                    y[DEBUG]: Atomically writing to file /var/lib/cloud/data/status.json (via ... Show more
2021-02-06    # _time: 1612592951.841
00:29:11.841  α az: a
-06:00        α cribl_breaker: Break on newlines
              α cribl_pipe: setting_index_by_region_availability_zone
              α domain: compute.amazonaws.com
              α host: ec2-21-187-232-201
              α region: asia-northeast3
```

Lookups

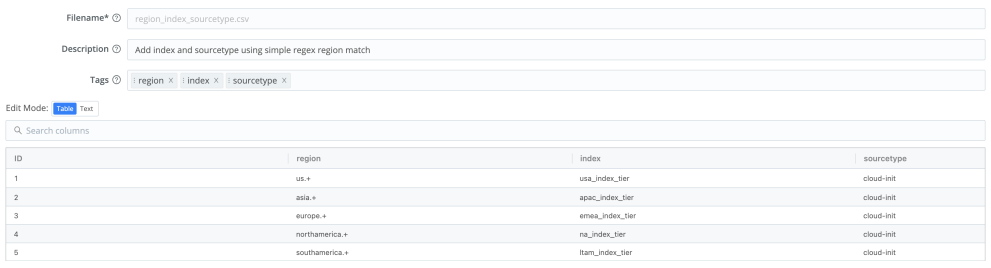
We still need to determine the index and sourcetype. Cribl Stream's Lookup Function enriches events with external fields. We'll use it with the newly extracted `region` field to assign an `index` and `sourcetype` to these events.

Lookup File

First, we need to create a lookup table for the Function to reference. For this, we'll use regex again.

In the table below, five simple regular expressions map the extracted `region` field to the appropriate `index` and `sourcetype`. For example, the region `us-west1-a` starts with `us`, so it matches the first regular expression: `us.+`

We use this lookup table's first row to assign an index of `usa_index_tier`, and a sourcetype of `cloud-init`, to each matching event. The region patterns in the table's four remaining rows work the same way.



Filename* region_index_sourcetype.csv

Description Add index and sourcetype using simple regex region match

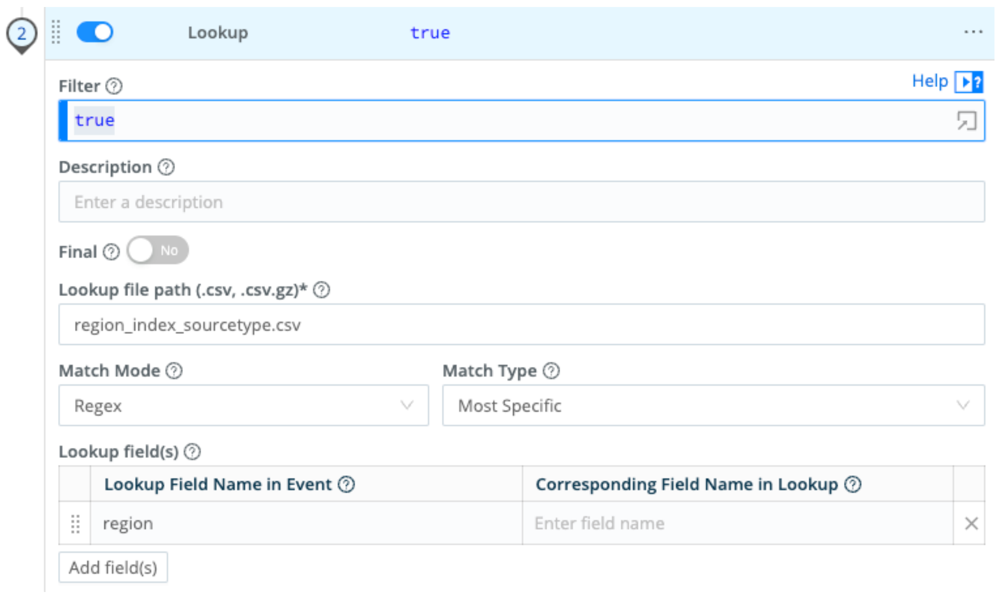
Tags region index sourcetype

Edit Mode: Table Text

ID	region	index	sourcetype
1	us.+	usa_index_tier	cloud-init
2	asia.+	apac_index_tier	cloud-init
3	europe.+	emea_index_tier	cloud-init
4	northamerica.+	na_index_tier	cloud-init
5	southamerica.+	ltam_index_tier	cloud-init

Lookup Function

With the lookup table saved as `region_index_sourcetype.csv`, the Lookup Function below matches the events' extracted `region` field against the regular expressions, and returns the matching `index` and `sourcetype`.



2 Lookup true

Filter: `true`

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: region_index_sourcetype.csv

Match Mode: Regex

Match Type: Most Specific

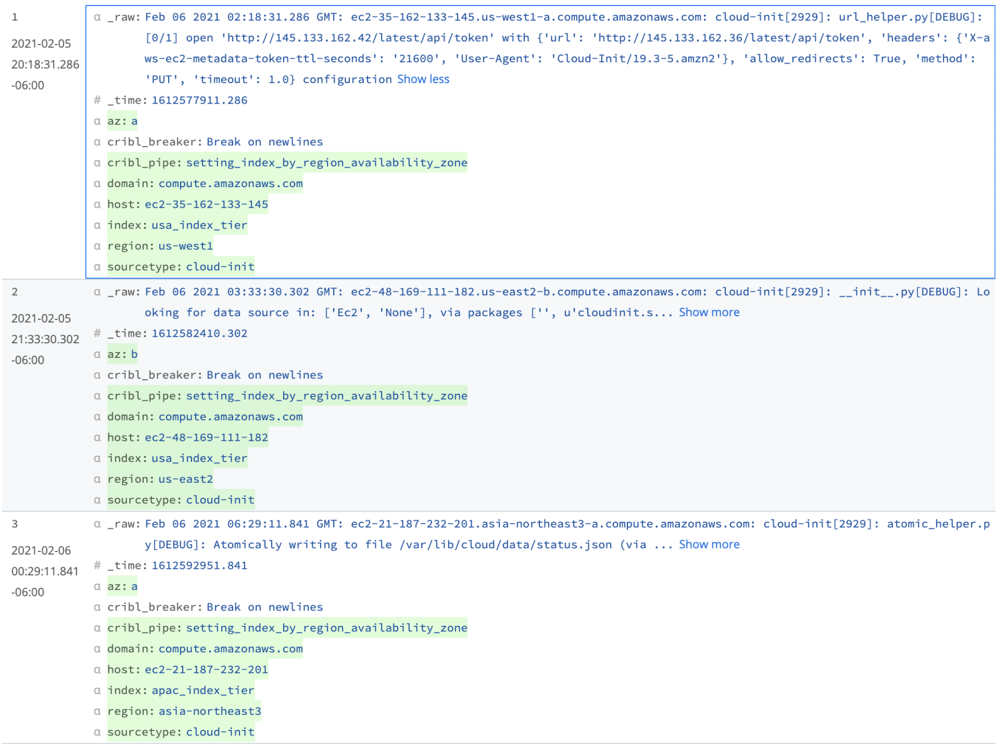
Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
region	Enter field name

There's actually more here than meets the eye. Note that we've specified no **Output Fields**. From the Lookup Function's documentation, we know this means that the Function will default to outputting **all** fields in the lookup table. So we get the contents of both remaining columns in the table we saw above: `index` and `sourcetype`.

Results of the Lookup Function

With the Lookup Function added to our Pipeline, the Preview pane's **OUT** tab shows that the `index` and `sourcetype` are now added to each event.



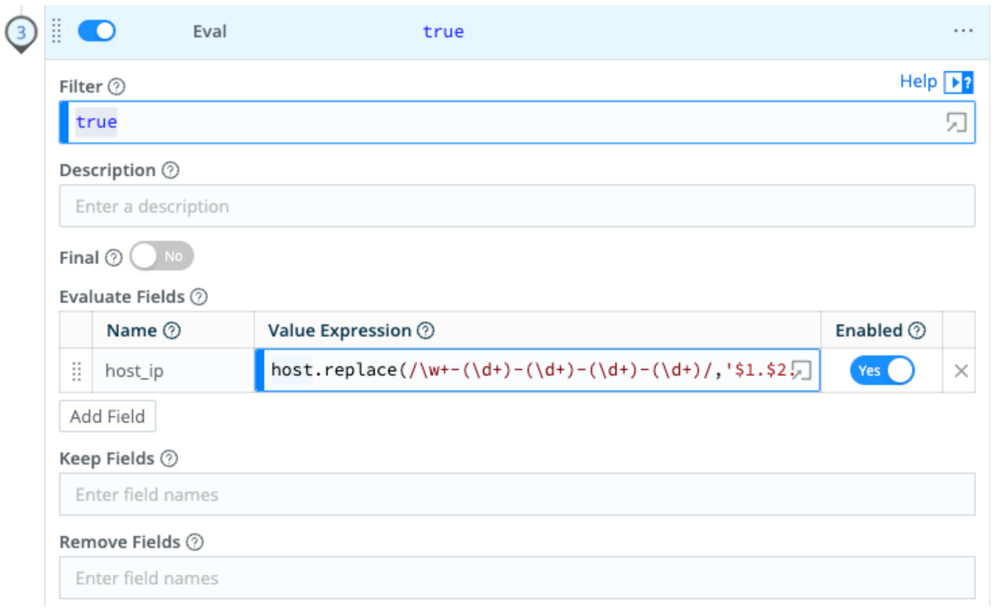
```
1            α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
                     [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://145.133.162.36/latest/api/token', 'headers': {'X-a
2021-02-05           ws-ec2-metadata-token-ttl-seconds': '21600', 'User-Agent': 'Cloud-Init/19.3-5.amzn2'}, 'allow_redirects': True, 'method':
20:18:31.286         'PUT', 'timeout': 1.0} configuration Show less
-06:00       # _time: 1612577911.286
             α az: a
             α cribl_breaker: Break on newlines
             α cribl_pipe: setting_index_by_region_availability_zone
             α domain: compute.amazonaws.com
             α host: ec2-35-162-133-145
             α index: usa_index_tier
             α region: us-west1
             α sourcetype: cloud-init
2            α _raw: Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com: cloud-init[2929]: __init__.py[DEBUG]: Lo
                     oking for data source in: ['Ec2', 'None'], via packages ['', u'cloudinit.s... Show more
2021-02-05   # _time: 1612582410.302
21:33:30.302 α az: b
-06:00       α cribl_breaker: Break on newlines
             α cribl_pipe: setting_index_by_region_availability_zone
             α domain: compute.amazonaws.com
             α host: ec2-48-169-111-182
             α index: usa_index_tier
             α region: us-east2
             α sourcetype: cloud-init
3            α _raw: Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaws.com: cloud-init[2929]: atomic_helper.p
                     y[DEBUG]: Atomically writing to file /var/lib/cloud/data/status.json (via ... Show more
2021-02-06   # _time: 1612592951.841
00:29:11.841 α az: a
-06:00       α cribl_breaker: Break on newlines
             α cribl_pipe: setting_index_by_region_availability_zone
             α domain: compute.amazonaws.com
             α host: ec2-21-187-232-201
             α index: apac_index_tier
             α region: asia-northeast3
             α sourcetype: cloud-init
```

Getting Host IP Address from Host Name

Since the IP address is present in the `host` field, we can create the `host_ip` field using an Eval Function with this replace method:

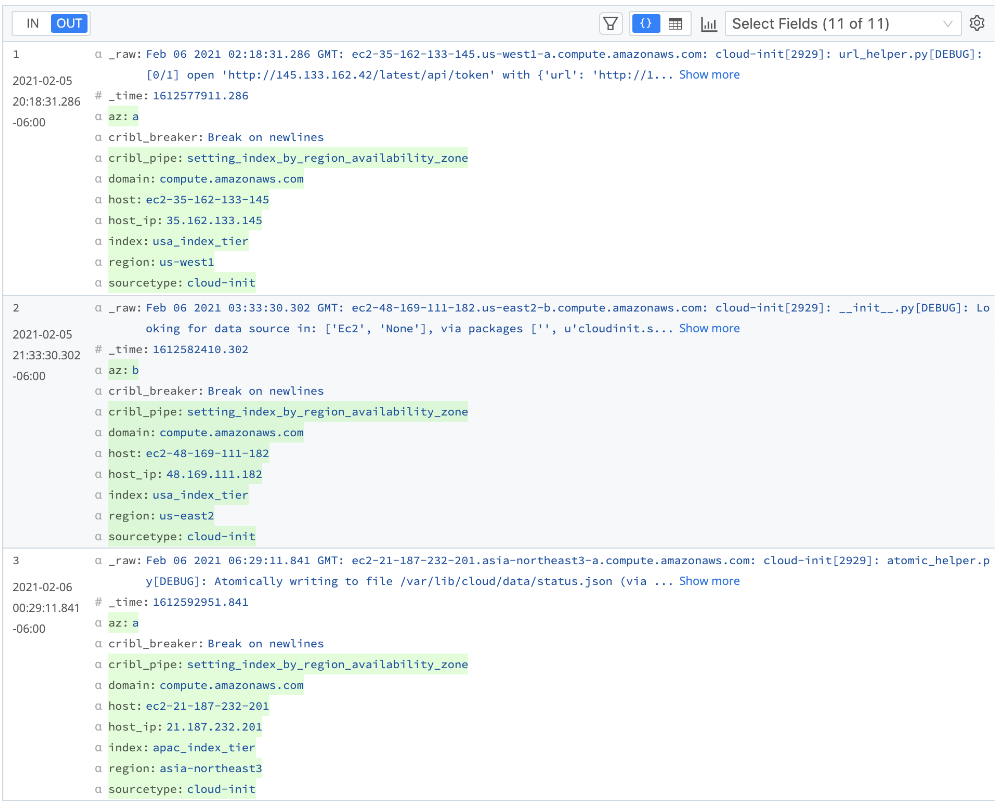
```
host.replace(/\w+-(\d+)-(\d+)-(\d+)-(\d+)/,'$1.$2.$3.$4')
```

This regular expression uses capture groups and pulls the four IP octets present in the hostname to build the `host_ip`. These four capture groups are noted as `$1.$2.$3.$4`, respectively. This method is very fast, and removes the need to perform a DNS lookup from the `host` field to get the host's IP address. Magic!



Results of the Eval Function and Replace Method

The `host_ip` field is now added to the events, displayed below `host`:

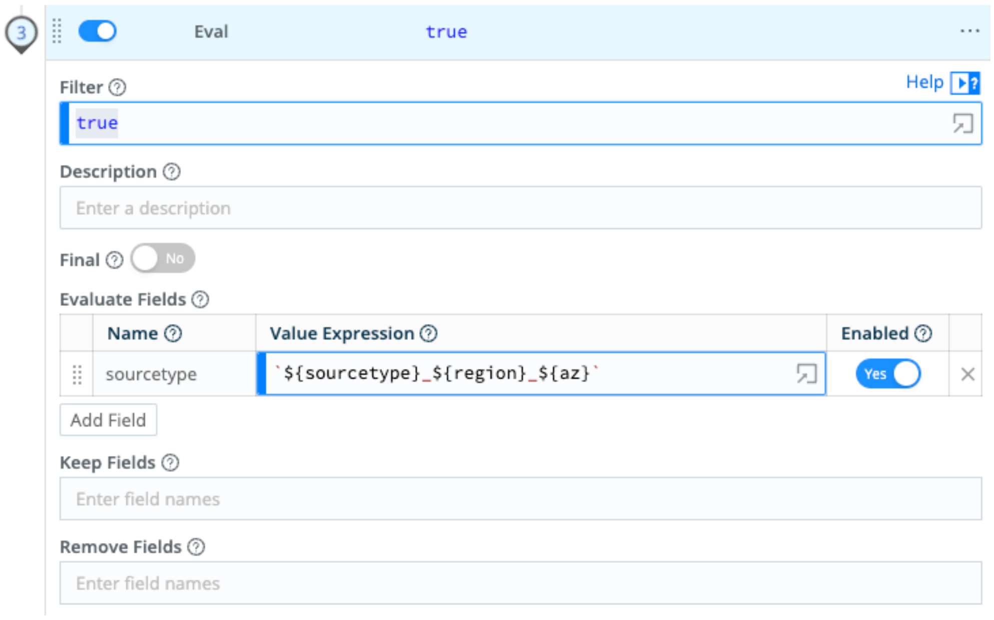


```
1                 _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
                        [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
2021-02-05        _time: 1612577911.286
20:18:31.286      az: a
-06:00            cribl_breaker: Break on newlines
                  cribl_pipe: setting_index_by_region_availability_zone
                  domain: compute.amazonaws.com
                  host: ec2-35-162-133-145
                  host_ip: 35.162.133.145
                  index: usa_index_tier
                  region: us-west1
                  sourcetype: cloud-init
2                 _raw: Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com: cloud-init[2929]: __init__.py[DEBUG]: Lo
                        oking for data source in: ['Ec2', 'None'], via packages ['', u'cloudinit.s... Show more
2021-02-05        _time: 1612582410.302
21:33:30.302      az: b
-06:00            cribl_breaker: Break on newlines
                  cribl_pipe: setting_index_by_region_availability_zone
                  domain: compute.amazonaws.com
                  host: ec2-48-169-111-182
                  host_ip: 48.169.111.182
                  index: usa_index_tier
                  region: us-east2
                  sourcetype: cloud-init
3                 _raw: Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaws.com: cloud-init[2929]: atomic_helper.p
                        y[DEBUG]: Atomically writing to file /var/lib/cloud/data/status.json (via ... Show more
2021-02-06        _time: 1612592951.841
00:29:11.841      az: a
-06:00            cribl_breaker: Break on newlines
                  cribl_pipe: setting_index_by_region_availability_zone
                  domain: compute.amazonaws.com
                  host: ec2-21-187-232-201
                  host_ip: 21.187.232.201
                  index: apac_index_tier
                  region: asia-northeast3
                  sourcetype: cloud-init
```

Customizing the Sourcetype

Finally, let's put some sense into the `sourcetype` field, using another Eval Function. By combining the values of the `${sourcetype}_${region}_${az}`, the sourcetype becomes `cloud-init_us-west1_a` – so now you can understand much more about the sourcetype at a glance.

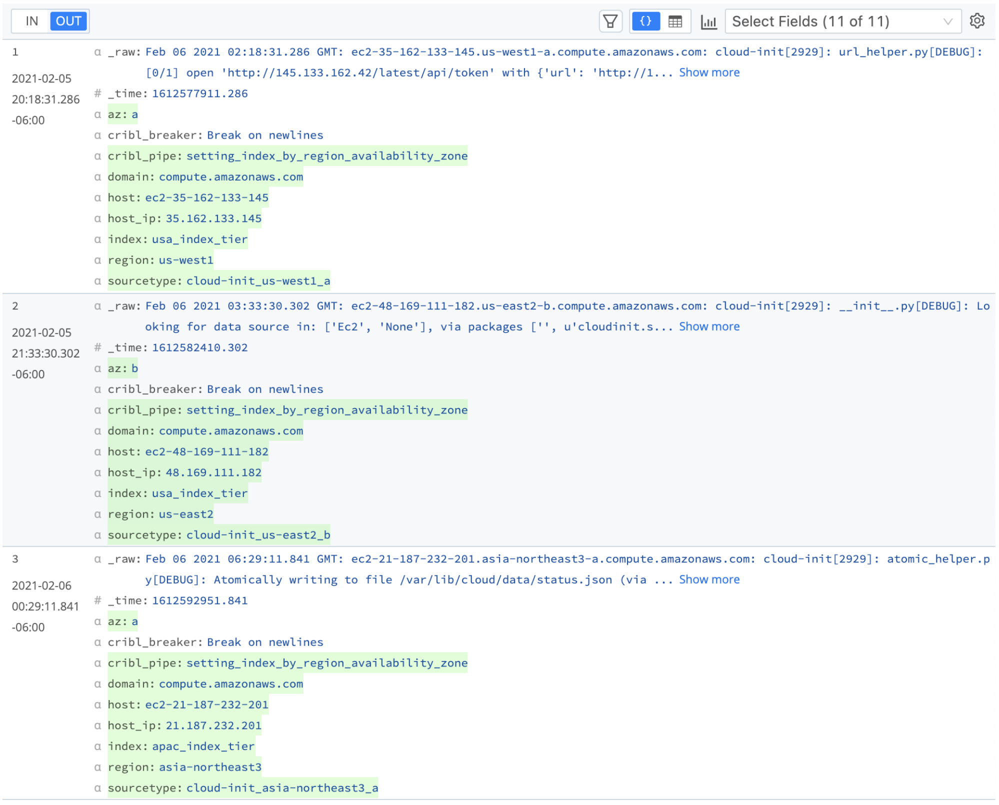
Examine this Eval Function's value expression, taking careful note of the backticks (` `) and braces ({ }) that surround the field names, and the underscore (_) that separates them.



Name	Value Expression	Enabled
sourcetype	`` `${sourcetype}_${region}_${az}` ``	Yes

Results of the Eval Function to Combine Values

Take a look at the updated sourcetypes, and enjoy exploring Cribl Stream with your new knowledge!



```
1             _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
2021-02-05          [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
20:18:31.286  _time: 1612577911.286
-06:00        az: a
              cribl_breaker: Break on newlines
              cribl_pipe: setting_index_by_region_availability_zone
              domain: compute.amazonaws.com
              host: ec2-35-162-133-145
              host_ip: 35.162.133.145
              index: usa_index_tier
              region: us-west1
              sourcetype: cloud-init_us-west1_a
2             _raw: Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com: cloud-init[2929]: __init__.py[DEBUG]: Lo
2021-02-05          oking for data source in: ['Ec2', 'None'], via packages ['', u'cloudinit.s... Show more
21:33:30.302  _time: 1612582410.302
-06:00        az: b
              cribl_breaker: Break on newlines
              cribl_pipe: setting_index_by_region_availability_zone
              domain: compute.amazonaws.com
              host: ec2-48-169-111-182
              host_ip: 48.169.111.182
              index: usa_index_tier
              region: us-east2
              sourcetype: cloud-init_us-east2_b
3             _raw: Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaws.com: cloud-init[2929]: atomic_helper.p
2021-02-06          y[DEBUG]: Atomically writing to file /var/lib/cloud/data/status.json (via ... Show more
00:29:11.841  _time: 1612592951.841
-06:00        az: a
              cribl_breaker: Break on newlines
              cribl_pipe: setting_index_by_region_availability_zone
              domain: compute.amazonaws.com
              host: ec2-21-187-232-201
              host_ip: 21.187.232.201
              index: apac_index_tier
              region: asia-northeast3
              sourcetype: cloud-init_asia-northeast3_a
```

Try This at Home

Below you'll find the lookup table, Pipeline, and sample events demonstrated in this use case. Create the lookup file first, and then import the Pipeline. (The order matters, because the Pipeline import depends on the lookup table's presence.)

Lookup Table

To create the lookup table in Cribl Stream's UI, select **Knowledge** > **Lookups**, then click **New Lookup File** and select **Create with Text Editor**.

Copy and paste in the header and rows listed below, then save the result as: `region_index_sourcetype.csv`.

region_index_sourcetype.csv

```
region,index,sourcetype
us.+,usa_index_tier,cloud-init
asia.+,apac_index_tier,cloud-init
europe.+,emea_index_tier,cloud-init
northamerica.+,na_index_tier,cloud-init
southamerica.+,ltam_index_tier,cloud-init
```

Pipeline

Below is an export of the whole Cribl Stream Pipeline presented here. Import this JSON to get a Pipeline named: `setting_index_by_region_availability_zone.json`.

Magic Pipeline

```
{
  "id": "setting_index_by_region_availability_zone",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "filter": "true",
        "conf": {
          "comment": "This pipeline demonstrates four different ways to leverage r
        },
        "id": "comment"
      },
      {
        "filter": "true",
        "conf": {
          "source": "_raw",
          "iterations": 100,
          "overwrite": false,
          "regex": "/GMT:\\s+(?<host>[^.]+)\\.(?<region>\\w+-\\w+\\d+)-(?<az>[^.]+
        },
        "id": "regex_extract",
        "disabled": false,
        "description": "Extract host, region, availability_zone, and domain"
      },
      {
        "filter": "true",
        "conf": {
          "matchMode": "regex",
          "matchType": "specific",
          "reloadPeriodSec": 60,
          "addToEvent": false,
          "inFields": [
            {
              "eventField": "region"
            }
          ],
          "ignoreCase": false,
          "file": "region_index_sourcetype.csv"
        },
        "id": "lookup",
        "disabled": false,
        "description": "Lookup index and sourcetype using regex matching"
      },
      {
        "filter": "true",
        "conf": {
          "add": [
            {
              "name": "host_ip",
              "value": "host.replace(/\\w+-(\\d+)-(\\d+)-(\\d+)-(\\d+)/,'$1.$2.$3.
            }
          ]
        },
        "id": "eval",
        "description": "Create IP from hostname, removing DNS requirement",
        "disabled": false
      },
      {
        "filter": "true",
        "conf": {
          "add": [
            {
              "name": "sourcetype",
              "value": "`${sourcetype}_${region}_${az}`"
            }
          ]
        },
        "id": "eval",
        "description": "Append region and az to sourcetype",
        "disabled": false
      }
    ]
  }
}
```

Sample Events

And here's a sample of raw events that you can upload or copy/paste into Cribl Stream's Preview pane to test the Pipeline's Functions:

Sample events

```
  Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com:
  Feb 06 2021 03:33:30.302 GMT: ec2-48-169-111-182.us-east2-b.compute.amazonaws.com:
  Feb 06 2021 06:29:11.841 GMT: ec2-21-187-232-201.asia-northeast3-a.compute.amazonaw
  Feb 06 2021 12:59:44.232 GMT: ec2-76-187-246-132.europe-west3-b.compute.amazonaws.
  Feb 06 2021 17:04:16.921 GMT: ec2-67-205-202-104.northamerica-northeast1-c.compute.
  Feb 06 2021 19:45:47.687 GMT: ec2-87-209-176-201.southamerica-east1-a.compute.amaz
```

From here, modify the sample data, lookup table, and Functions to adapt this approach to your own needs!

20.2.5. Updating Lookups In Multiple Packs

Lookups provide a flexible way to enrich events en route to their Destinations. You can use Lookups in multiple Packs to share complex configurations and workflows across multiple Worker Groups, or across organizations.

Here, we will be walking you through a particularly challenging setup: What happens if you have a very large Lookup that requires frequent updates? To make it more interesting, what happens if the same Lookup is being used in multiple Packs? In the first part, we will discuss how to update a Lookup file in multiple Packs and the second, we will address the best way to set up a large Lookup file that needs frequent updates.

Updating Lookups in Multiple Packs

Manually updating multiple Packs with an ever-evolving Lookup table is a non-starter, so let's examine a better alternative.

Create the Lookup File in the Worker Group

The first step is to create a Lookup file in the Worker Group. For ease of sharing and use, make sure the Lookup file is not located in the Pack.

Our example Lookup file is `host_destination_test.csv`.

Validate that the Lookup File Is on the Worker Node

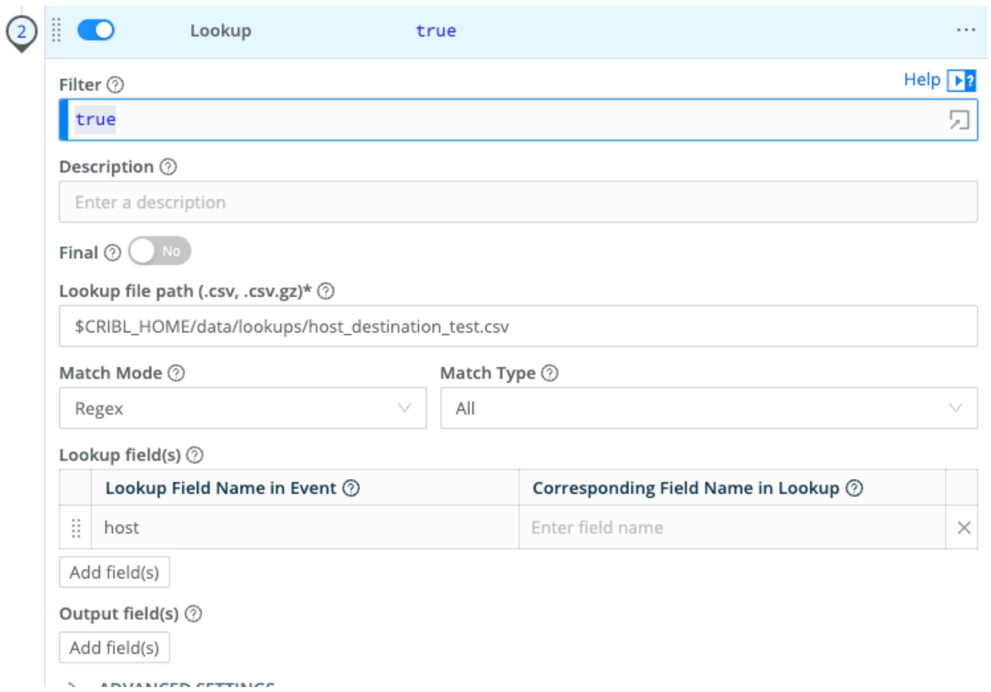
After you **Commit** and **Deploy** to propagate the Lookup table to the Workers, validate that you can view the Lookup.

In our example, the path is: `$CRIBL_HOME/data/lookups/host_destination_test.csv`.

Point the Pack(s) to the Lookup File

In your Pack(s), add a Lookup Function and point it to the Worker Node Lookup file location. In our example, the path is: `$CRIBL_HOME/data/lookups/host_destination_test.csv`.

Do not use point the Pack to the Leader Lookup File location, such as: `$CRIBL_HOME/groups/<your group>/data/lookups/host_destination_test.csv`. Also, do not point it directly to the Pack Lookup file, such as `host_destination_test.csv`.


Lookup true

Filter: true

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: $CRIBL_HOME/data/lookups/host_destination_test.csv

Match Mode: Regex

Match Type: All

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
host	Enter field name

Output field(s)

Lookup Function's configuration

Setting up a Lookup File with Dynamic Updates

In this part, we will walk through the best way to set up a large Lookup file that requires frequent updates.

We will use the Cribl Lookups APIs to:

Update a Lookup file → Load it to the Leader Node's `$CRIBL_HOME/groups/<Group-Name>/data/lookups` subdirectory → Commit and Deploy → The changes in the Lookup file will propagate to the Worker Nodes that belong to this Worker Group.

The advantage of this approach, compared to updating the Lookup files on the Worker Nodes direcly, is that you do not need to know all the `IP` address and `hostnames` of the Worker Nodes. It's also more efficient to update in one location.

The sequence of the API calls are as follows:

- `PUT` the Lookup into the Leader Node's `$CRIBL_HOME/groups/<Group-Name>/data/lookups` subdirectory.
- `PATCH` the Lookup (with updates) into Leader Node's `$CRIBL_HOME/groups/<Group-Name>/data/lookups` subdirectory.
- `POST` to commit and push the new Lookup table to the Worker Nodes in the Worker Group.
- `PATCH` to deploy the changes.

Store the Lookup Table on the Leader Node

To set up a Lookup table that can be updated dynamically, upload it and store it on the Leader Node. In our example, we stored the file in: `/opt/host_destinations.csv`.

PUT the Lookup into the Worker Group

In this step, we use the Cribl Lookup API to `PUT` the Lookup into the Worker Group. Run the following in your terminal:

```
curl -X PUT "http://192.168.56.254:9000/api/v1/m/default/system/lookups?filename=ho
-H "Authorization: Bearer <your token>" -H 'Content-Type: text/csv'
--data-binary '@/opt/host_destination.csv'
```

Response:

```
filename":"host_destination.csv.kmtVCQs.tmp","rows":11,"size":245
```

PATCH the Lookup

Using the Cribl Lookup API, we use the temporary file `host_destination.csv.kmtVCQs.tmp` from the prior step as the input to our `PATCH` command:

```
curl -X PATCH "http://192.168.56.254:9000/api/v1/m/default/system/lookups/host_dest
-H "accept: application/json" -H "Authorization: Bearer <your token>"
-H "Content-Type: application/json"
-d '{"id":"host_destination.csv","fileInfo":{"filename":"host_destination.csv.kmtV
```

Response:

```
{"items":[{"id":"host_destination.csv","size":245}],"count":1}
```

Commit the Lookup to the Worker Group

In our example, we are using the `default` Worker Group. We are also using the API selectively to commit only the Lookup table to the `default` Worker Group. For details, see Selectively Commit Changes.

```
  curl -X POST "http://192.168.56.254:9000/api/v1/version/commit"
  -H "accept: application/json" -H "Authorization: Bearer <your token>"
  -H "Content-Type: application/json"
  -d '{"message":"automation@cribl:commit","group":"default","files":["groups/defaul
```

Response:

```
{"items":
[{"branch":"master","commit":"8377ec8bbd21f460d9554f671dc79d217d991afd","root":false,"
{"changes":2,"insertions":2,"deletions":0},"files":{"modified":
["groups/default/data/lookups/host_destination.csv","groups/default/data/lookups/host_
```

Save the commit number in the response payload for the next step: `8377ec8bbd21f460d9554f671dc79d217d991afd`

Deploy the Lookup to the Worker Nodes

In this final step, we use `PATCH` to deploy the commit to the Worker Nodes. Note the value of the commit from the previous step.

```
  curl -X PATCH "http://192.168.56.254:9000/api/v1/master/groups/default/deploy"
  -H "accept: application/json" -H "Authorization: Bearer <your token>"
  -H "Content-Type: application/json"
  -d '{"version":"8377ec8bbd21f460d9554f671dc79d217d991afd"}'
```

Response:

```
{"items":[{"description":"Default Worker
Group","tags":"default","configVersion":"8377ec8bbd21f460d9554f671dc79d217d991afd","wo
```

20.2.6. Updating Lookups With A REST API

This page documents a technique for uploading, committing, and deploying Lookup files with a REST API. In this use case, an external team modifies the Lookup file nightly. Our procedure describes a method for deploying the modified file to the Worker Groups.

As an alternate method, you can also use the Cribl API to upload a Lookup file.

1. Create the Lookup table.

Create a Lookup table and store it on the Leader Node. In this example, the path and filename are `/opt/host_destinations.csv`.

2. Send the PUT command to upload the modified Lookup file.

Note the location of the external updated CSV file: `@/opt/host_destination.csv`. Then send the following command:

```
curl -X PUT "http://192.168.56.254:9000/api/v1/m/default/system/lookups?filename=ho
-H "Authorization: Bearer <token> -H 'Content-Type: text/csv'
--data-binary '@/opt/host_destination.csv'
```

You'll receive a JSON object response like this:

```
{"filename":"host_destination.csv.kmtVCQs.tmp","rows":11,"size":245}
```

Bearer Tokens

See bearer token for information on incorporating your bearer token into the cURL command.

3. Send the PATCH command to update the Lookup file on the Worker Group.

We use the temporary file (`host_destination.csv.kmtVCQs.tmp`) from the `PUT` command as input to the `PATCH` command.

```
curl -X PATCH "http://192.168.56.254:9000/api/v1/m/default/system/lookups/host_dest
-H "accept: application/json" -H "Authorization: Bearer <token> -H "Content-Type: a
-d '{"id":"host_destination.csv","fileInfo":{"filename":"host_destination.csv.kmtVC
```

You'll receive a JSON object response like this:

```
{"items":[{"id":"host_destination.csv","size":245}],"count":1}
```

4. Send the POST command to commit the modified Lookup file.

In the following example, we use `default` as the Worker Group name.

Note the second value from the commit: `8377ec8bbd21f460d9554f671dc79d217d991afd`. We'll need it for the `deploy` command.

Note that the following command only commits the Lookup table in the `default` Worker Group. See our version control documentation for more information on selectively committing changes.

```
curl -X POST "http://192.168.56.254:9000/api/v1/version/commit"
-H "accept: application/json" -H "Authorization: Bearer <token> -H "Content-Type: a
-d '{"message":"automation@cribl:commit","group":"default","files":["groups/default
```

Response:

```
{"items":
[{"branch":"master","commit":"8377ec8bbd21f460d9554f671dc79d217d991afd","root":false,"
{"changes":2,"insertions":2,"deletions":0},"files":{"modified":
["groups/default/data/lookups/host_destination.csv","groups/default/data/lookups/host_
```

5. Send the PATCH command to deploy the modified Lookup file.

Once again, we use `default` as the Worker Group name.

Note the second value from the commit in the deploy version:
`8377ec8bbd21f460d9554f671dc79d217d991afd`.

```
curl -X PATCH "http://192.168.56.254:9000/api/v1/master/groups/default/deploy"
-H "accept: application/json" -H "Authorization: Bearer <token> -H "Content-Type: a
-d '{"version":"8377ec8bbd21f460d9554f671dc79d217d991afd"}'
```

Response:

```
{"items":[{"description":"Default Worker
Group","tags":"default","configVersion":"8377ec8bbd21f460d9554f671dc79d217d991afd","wo
```

20.3. Sampling Applications

20.3.1. Sampling

Sampling at Ingest-Time

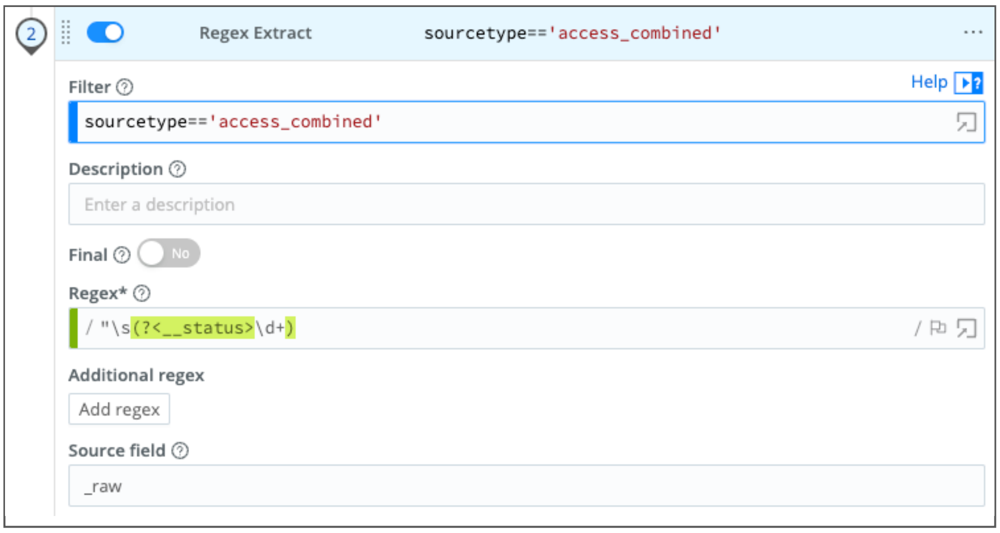
Let’s say that you wanted to analyze and troubleshoot with **highly verbose/voluminous** data – for example, CDN logs, ELB Access Logs, or VPC Flows – but you were concerned about storage requirements and search performance. With Sampling, you can bring in enough samples that your analysis remains statistically significant, and also do all the necessary troubleshooting.

See the example below, or see more details in Access Logs and Firewall Logs.

Sampling Example

Let’s use the out-of-the-box **Sampling** function to sample all events from `sourcetype=='access_combined'` where `status` is `'200'`. We’ll sample these at 5:1 (and all other events at 1:1). This should lower the volume of all verbose successes (`200`s), but still bring in **all** potentially erroneous events (`400`s, `500`s, etc.) that can be used for troubleshooting.

- First, make sure you have a Route and Pipeline configured to match desired events.
- Next, let’s add a **Regex Extract** Function to extract the status field from `_raw`, and let’s call the resulting field `__status`. Remember, fields that start with `__` are special fields in Cribl Stream, and can be used anywhere in a Pipeline.



Regex Extract `sourcetype=='access_combined'`

Filter: `sourcetype=='access_combined'`

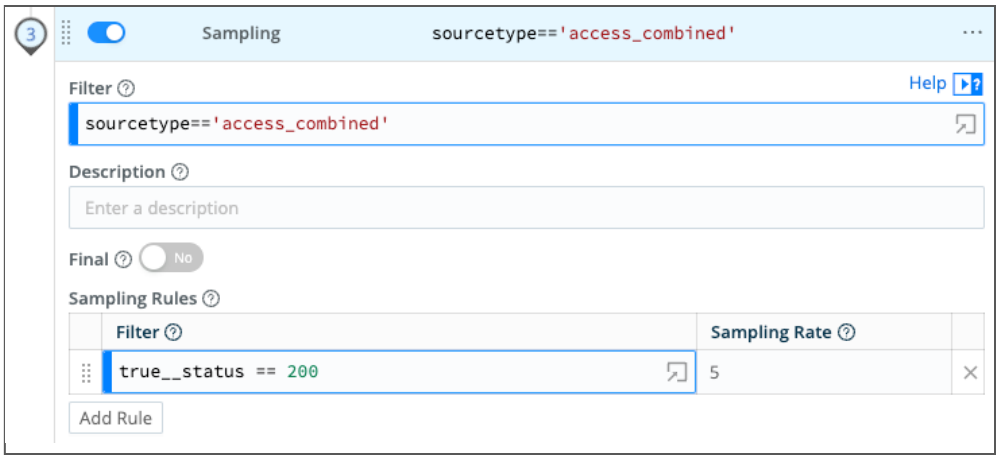
Description: Enter a description

Regex*: `/ "\s(?<__status>\d+)`

Source field: _raw

Extracting the `__status` field

Next, let's add a **Sampling** function, and scope it to all events where `sourcetype=='access_combined'`. Let's apply a filter condition of `__status == 200`, and a Sample Rate of `5`.



Sampling sourcetype=='access_combined'

Filter: sourcetype=='access_combined'

Description: Enter a description

Final: No

Sampling Rules

Filter	Sampling Rate
true__status == 200	5

Sampling success responses

To confirm that sampling works, compare the event count of all `200`s before and after.

Each time an event goes through the **Sampling** function, an index-time `sampled::<rate>` field is added to it. You can use this field in your statistical functions, as necessary.

20.3.2. Sample Logs

Collecting samples of the event data you plan to work with in Cribl Stream can make your Cribl Stream onboarding experience even quicker and more efficient than if you don't.

When you set out to collect sample data, first decide whether to collect before or after the sending agent processes it.

- If you can, collect the data **after** the agent processes it. Then you can set up Cribl Stream Pipelines more accurately the first time. The best option is live capture with Cribl Stream. Next best is gathering files.
- If the only practical option is to collect data **before** it reaches the agent, that works, too. (This will usually be a matter of gathering files.) Since setting up Pipelines is an iterative process, you can still arrive at an optimal setup even if it takes a bit longer.

Regardless of what sending agent you're using, either the Capturing Data with Cribl Stream or the Gathering Original Files section will apply.

Beyond that, there are two supplemental sections: one about how to export data from Elasticsearch, and the other about exporting from Splunk. Of course, these are only two possibilities in a long list of sending agents. If you are using a sending agent that's not covered here, please join us on Cribl's Community Slack at https://cribl-community.slack.com/ and share your questions about collecting sample logs.

Describing Sample Data

For every group of sample files, create a `README` file which includes:

- Sourcetype (e.g., the type of log you're collecting samples from).
- Originating app or appliance.
- Delivery mechanism (e.g., syslog, Filebeat, Splunk UF, or something else).
- Whether the was data collected before or after processing by the agent.
- A brief description of the data.

Capturing Data with Cribl Stream

Though not always the easiest, this is the best option, because samples are captured exactly as they will look when handed off to Cribl Stream in production. You'll set Cribl Stream up as a passthrough which gets samples "right off the wire" and dumps them to DevNull.

Configuring Sources

In Cribl.Cloud:

Check the landing page to see whether the pre-defined Sources (Elastic, syslog, and Splunk, among others) are sufficient for your needs. If you need others, configure them using ports in the range 20000-20010.

Or, if your deployment is on-prem or in a private cloud:

Download and install Cribl Stream.

```
$ mkdir /opt; cd /opt
$ curl -Lso - $(curl https://cdn.cribl.io/dl/latest-x64) | tar zxvf -
$ cribl/bin/cribl start
```

Open `localhost:9000` in a browser.

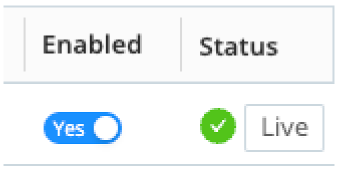
Configure the Source you need. For simplicity, use port 1024 or higher. (If you must use lower ports you'll need to follow a different installation procedure.)

Directing Data to Cribl Stream

Point your sending agent at the Cribl Stream Source you created above. How you do this depends on what sending agent you use. See this Cribl blog for a how-to that focuses on Cribl.Cloud but also applies to on-prem deployments.

Live-capturing Data

Once data has begun flowing, click the **Live** button for your chosen Source.



The **Live** button initiates capture

Grab 100 to 500 events, at a minimum. Try to capture as much variety as possible, even if that requires multiple captures.

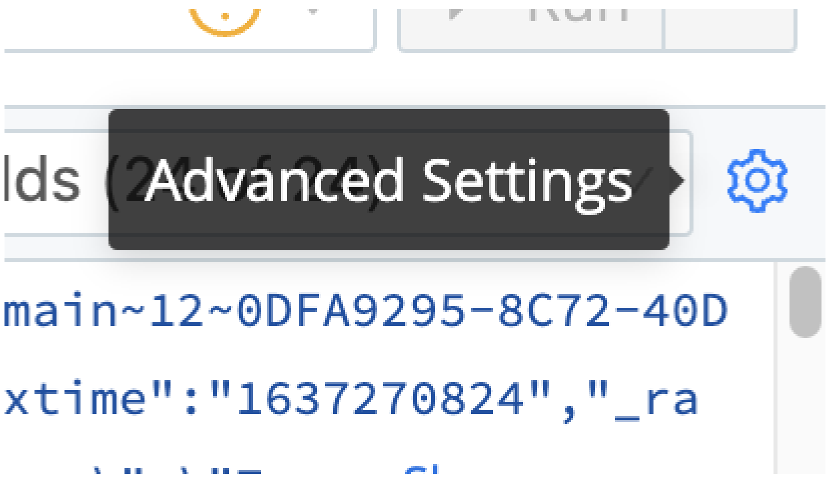
When finished, click **Save as Sample File**. Note the file name (and rename if you like). Close the sample window.

In the **Routing** > **Data Routes** tab, select the Route you are using.

Your sample file(s) should appear in the **Sample Data** pane on the right. If the **Sample Data** pane does not appear, type `]` (right square bracket) key to unhide it.

For each file you want to download:

- Click the file name.
- Click the (gear icon) for **Advanced Settings**:



The **Advanced Settings** icon

- In the resulting pop-up, select **Save** > **NDJSON**.
- Once the file finishes downloading:
    - Move the file into the directory you've created for that set of sample files - which should also include a `README` as described above.
- When all files have been moved into the directory, compress the directory as a `.tgz` archive.

Gathering Original Sample Files

If live-capturing data with Cribl Stream is impractical for you, gather sample files that the sending agent has not yet processed, using the following general workflow:

- Create a directory in which to store samples.
- Copy a sample file into the new directory.
- Repeat until all desired sample files have been copied.
- Add a `README` file to the directory, as described above.
- Redact sensitive content in the data, if required.

- Archive the directory (e.g., with a command like `tar -czf samples.tgz samples`) for portability; send it to your Cribl representative, if you are working with one.

Exporting Data from a Platform

If your data is in Elastic or Splunk, and you can neither capture live nor obtain original samples, use one of the following export procedures. Then complete the workflow above, using the exported files instead of original sample files.

If Splunk uses props and/or transforms to modify the `_raw` data of your logs, you **must** obtain your sample data from the source, because the exported data will **not** include events in their original form.

Exporting Data from Elastic

1. Elastic often transforms the data it stores. To the extent possible, note any particulars about the shape and content of the original data before it reached Elastic. Take special care to record the effects of any Logstash pipeline configs on the sample data. Record all this information in the `README` you create. This will make it easier to create the most effective filters in Cribl Stream.
2. Install the Logstash plugins logstash-input-elasticsearch and logstash-output-csv:

```
$ cd /opt/logstash
$ bin/logstash-plugin install logstash-input-csv
$ bin/logstash-plugin install logstash-output-csv
```

3. Compose a query to retrieve the data you want. Be sure to include the `message` field, whose value should be the original event.
4. Substitute your query for the placeholder in the following config snippet, and save the snippet as `output-csv.conf`:

```
input {
 elasticsearch {
    hosts => "elastic-host:9200"
    index => "target-index"
    query => '
  {"query": {
    # your ES query here
        }
      }
    }
}}'
  }
}
output {
  csv {
    fields => ["message", "field1", "field2"]
path => "/home/user/ES-sourcetype-export.csv"
  }
}
```

5. Run your export operation:

```
/opt/logstash/bin/logstash -f output-csv.conf
```

Exporting Data from Splunk

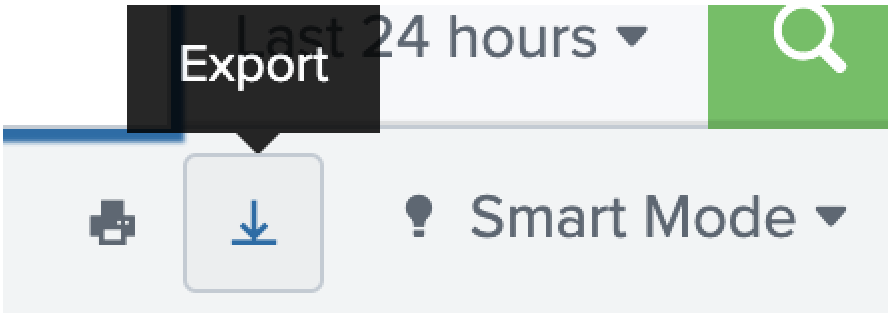
You can export logs from Splunk using the GUI or command line.

In the Splunk GUI:

1. Run a search that returns the appropriate sample of target logs. Adjust the number of samples by appending `| head <number_of_samples_desired>` to your search. For example, to grab 100 samples of Cisco ASA firewall data:

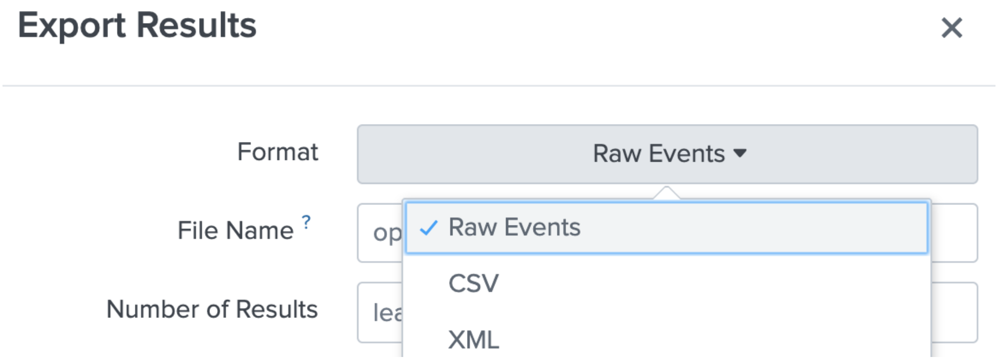
```
index=firewall sourcetype=cisco:asa | head 100
```

2. Adjust the search as needed to exclude or include the events you want to send.
3. When your search is collecting the desired samples, export the search results:



Exporting search results

4. Then select **Raw Events**, name the file after the source or sourcetype, and save.


Export Results

Format: Raw Events

File Name: op

Number of Results: lea

Exporting raw events

Or, in the Splunk CLI:

You can obtain the same results as described in the previous section using commands like the following.

```
% $SPLUNK_HOME/bin/splunk search "index=firewall sourcetype=cisco:asa | head 100" -
```

20.3.3. Access Logs: Apache, ELB, CDN, S3, Etc.

Recipe for Sampling Access Logs

Access logs are extremely common. They're often emitted by web servers or similar/related technologies (proxies, loadbalancers, etc.), and tend to be highly voluminous. Typical examples include Apache access logs, and CDN logs such as those from Amazon Cloudfront, Amazon S3 Server Access Logs, AWS ELB Access Logs, etc.

For large installations, bringing everything into an analytics tool is often so cost-prohibitive (storage, resources, license, etc.) that most users don't even bother. However, some of the logs contain relevant information when looked at individually (e.g., errors). The much larger majority contains relevant information when looked at in the aggregate (e.g., successes to determine traffic patterns, etc.).

It would be great if we could find a middle ground. With the Sampling Function, you can! Specifically, you can:

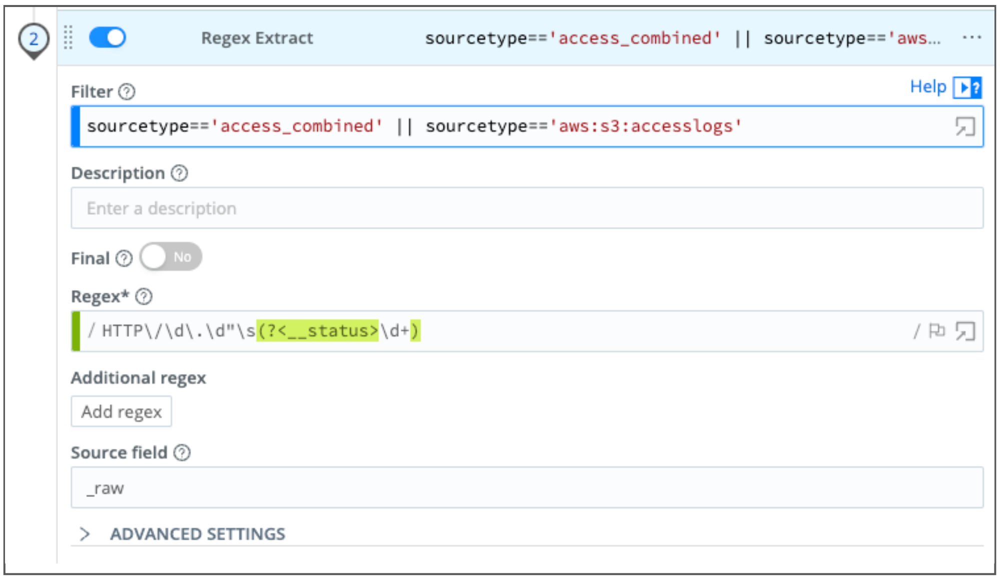
- Ingest enough sample events from the majority category that your aggregate analysis remains statistically significant.
- Ingest *all* events from the minority categories, and perform troubleshooting and introspection with full-fidelity data.

Using `status` as the Sampling Condition

Most of the access logs (including the ones mentioned above) have very similar formats. One quick way to sample is to look at the value of the `status` field. `2XX`s indicate success and tend to be, by far, the most common ones – with `200` being the top. **Therefore, `200` is the perfect candidate for sampling.** All other statuses occur much less frequently, indicate conditions that often need to be looked at, and can be brought in with full fidelity.

Sample Status 200 at 5:1

1. Add a **Regex Extract** Function that looks at these sourcetypes: `sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'`
2. Configure that Function to extract a field called `__status` with this regex: `/HTTP\/\d\.\d"\s(?<__status>\d+)/`


Regex Extract sourcetype=='access_combined' || sourcetype=='aws...

Filter: `sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'`

Description: Enter a description

Final: No

Regex*: `/ HTTP\/\d\.\d"\s(?<__status>\d+)`

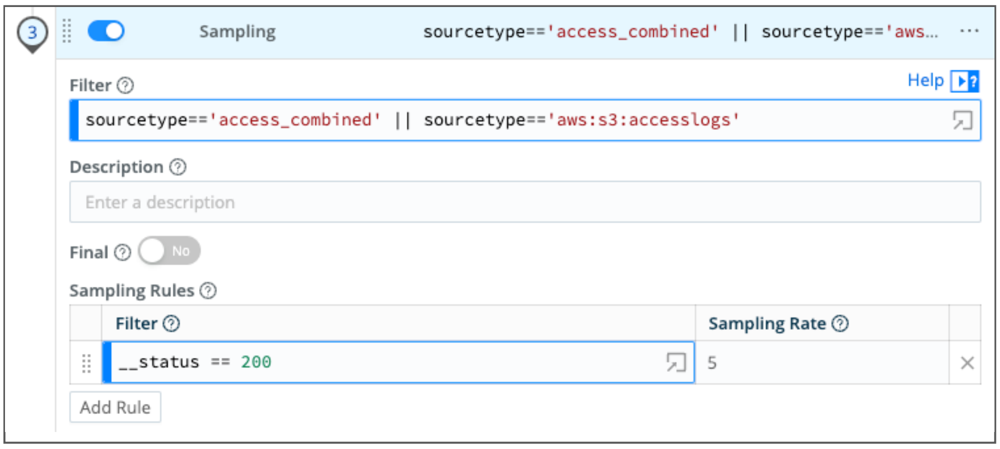
Additional regex: Add regex

Source field: `_raw`

ADVANCED SETTINGS

Defining the Regex Extract Function

3. Add a Sampling Function to sample `5:1` when `__status==200`.
4. Save.


Sampling sourcetype=='access_combined' || sourcetype=='aws...

Filter: `sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'`

Description: Enter a description

Final: No

Sampling Rules

Filter	Sampling Rate
`__status == 200`	5

Add Rule

Sampling success reponses

Note About Sampling

Each time an event goes through the **Sampling** Function, an index-time `sampled::<rate>` field is added to it. Use this field in your statistical Functions, as necessary.

Other Sourcetypes

Examples of other sourcetypes that will benefit from sampling, but might need a different `__status` extraction regex:

Sourcetype	Filter Expression
Amazon Cloudfront Access Logs	`sourcetype=='aws:cloudfront:accesslogs'`
Amazon ELB Access Logs	`sourcetype=='aws:elb:accesslogs'`

20.3.4. Firewall Logs: VPC Flow Logs, Cisco ASA, Etc.

Recipe for Sampling Firewall Logs

Firewall logs are another source of important operational (and security) data. Typical examples include Amazon VPC Flow Logs, Cisco ASA Logs, and other technologies such as Juniper, Checkpoint, pfSense, etc.

As with Access Logs, bringing in **everything** for operational analysis might be cost-prohibitive. But sampling with Cribl Stream can help you:

- Ingest enough sample events from the majority category that your aggregate analysis remains statistically significant. E.g., sample all `ACCEPT`s at `5:1`.
- Ingest **all** events from the minority categories, and perform troubleshooting and introspection with full-fidelity data. E.g., bring in all `REJECT`s.

Sampling VPC Flow Logs

AWS' VPC Flow Logs feature enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow Log data can be published to Amazon CloudWatch Logs and Amazon S3.

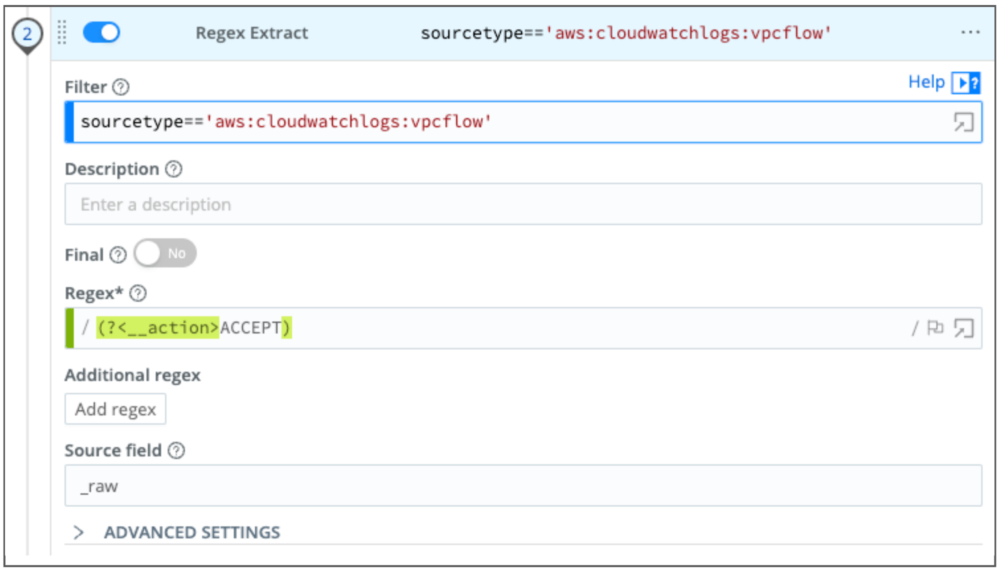
Typical VPC Flow Logs look like this:

Flow Log Records for Accepted and Rejected Traffic

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 14185300
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 14185300
```

Let's use a **very simple** Filter condition and only look for `ACCEPT` events:

1. Add a **Regex Extract** Function that looks at: `sourcetype=='aws:cloudwatchlogs:vpcflow'`
2. Configure that Function to extract a field called `__action` with this regex: `/(?<__action>ACCEPT)/`


Regex Extract sourcetype=='aws:cloudwatchlogs:vpcflow'

Filter: sourcetype=='aws:cloudwatchlogs:vpcflow'

Description: Enter a description

Final: No

Regex*: /(?<__action>ACCEPT)/

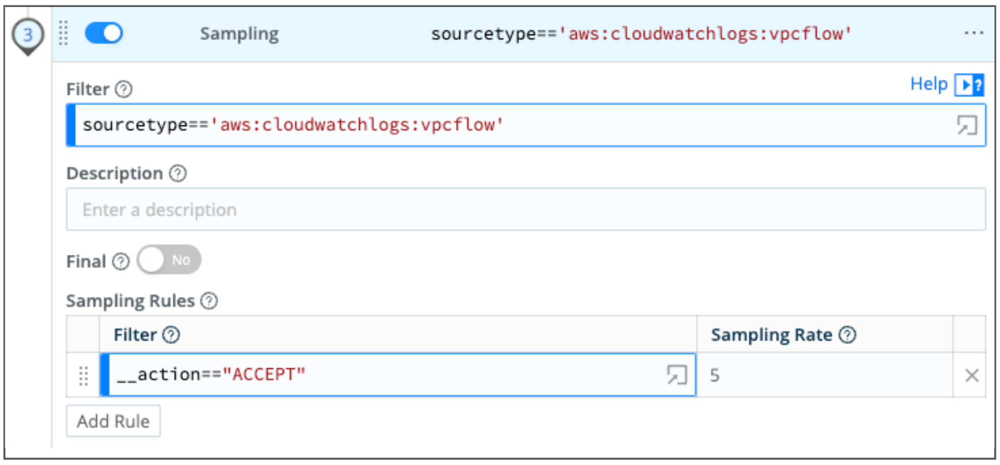
Additional regex: Add regex

Source field: _raw

ADVANCED SETTINGS

Extracting the `__action` field

3. Add a **Sampling** Function to sample `5:1` when `__action=="ACCEPT"`.
4. Save.


Sampling sourcetype=='aws:cloudwatchlogs:vpcflow'

Filter: sourcetype=='aws:cloudwatchlogs:vpcflow'

Description: Enter a description

Final: No

Sampling Rules

Filter	Sampling Rate
__action=="ACCEPT"	5

Add Rule

Sampling `ACCEPT` events

Note About Sampling

Each time an event goes through the Sampling Function, an index-time field is added to it: `sampled: <rate>`. It's advisable that you use that in your statistical functions, as necessary.

Other Sourcetypes

Other sourcetypes that will benefit from sampling, but might need a different `__action` extraction regex:

Sourcetype	Filter Expression
Cisco ASA Logs	`sourcetype=='cisco:asa'`

Related sourcetypes to consider sampling:	`sourcetype=='cisco:fwsm'`
	`sourcetype=='cisco:pix'`

20.4. Transformation Applications

20.4.1. Code Function Examples

The Code Function is a powerful way to transform events without needing to write a custom function. Using JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every` is possible.

Cribl still recommends that you use Cribl Stream's basic, built-in Functions, like Eval, as much as possible. But these use cases demonstrate some basic ways to use the new Code Function to solve questions asked in the Cribl's Slack Community.

Basic Examples

Example Event Data

The first several examples below use the following JSON object as a sample event. You can copy and paste this event into Cribl Stream's **Sample Data** pane. Add the `Do Not Break` Ruleset to your Source, select the `noBreak1MB` Event Breaker, and under your Source's **Advanced Settings**, enable `Parse JSON Event`.

```
{
  "cpus": [
    {"number": 1, "name": "CPU1", "value": 2.3},
    {"number": 2, "name": "CPU2", "value": 3.1},
    {"number": 3, "name": "CPU3", "value": 5.1},
    {"number": 4, "name": "CPU4", "value": 1.3}
  ],
  "arch": "Intel x64"
}
```

Accessing Fields in an Event

To access a field inside an event, you can use the `__e` special variable. The `__e` prefix allows for access to the `(context)` event inside a JavaScript expression. For example, if you want to access the extracted field `field-name`, use the following syntax:

```
__e['field-name']
```

In other words, think of your code executing in a context like this:

```
function(__e: Event) {
 // your code here
}
```

Eval a Field

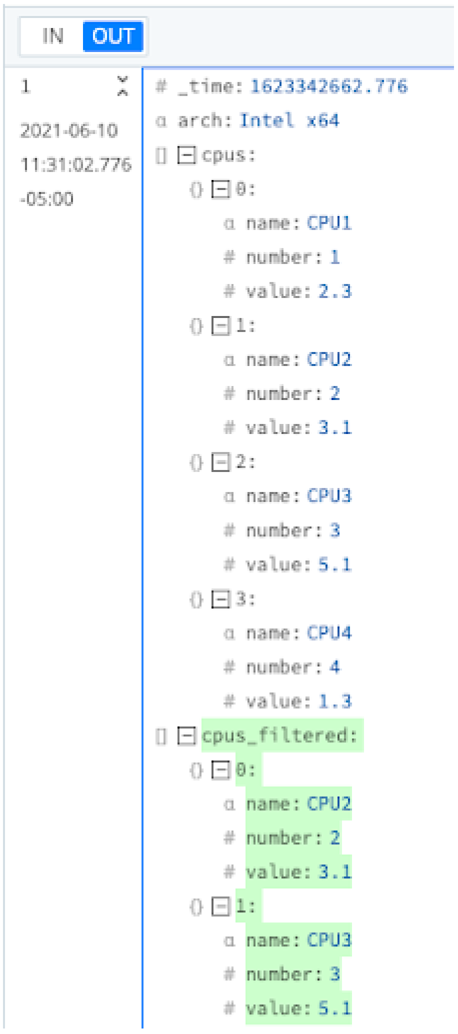
To create a new field, it is as simple as assigning the field to a value. For example, if you want to create a field `test` with the value `Hello, Goats!`, use the following syntax:

```
__e['test'] = 'Hello, Goats!'
```

JSON Filter

To filter the `cpus` array inside the event, you can use the `filter` function to keep only certain values, based on a logic condition. In the example below, only entries where the value is greater than or equal to `3` are kept, and placed into a new field called `cpus_filtered`.

```
__e['cpus_filtered'] = __e['cpus'].filter(entry => entry.value >= 3)
```



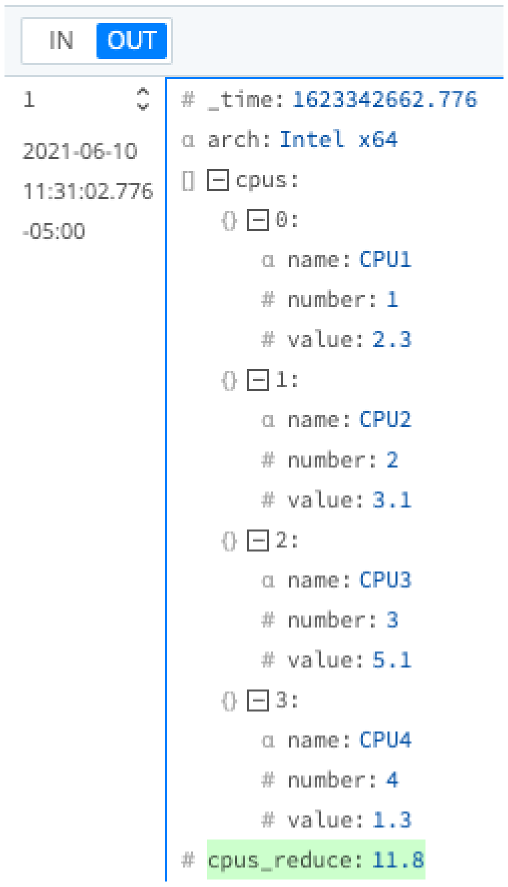
```
1
2021-06-10
11:31:02.776
-05:00

# _time: 1623342662.776
a arch: Intel x64
[] cpus:
  {} 0:
    a name: CPU1
    # number: 1
    # value: 2.3
  {} 1:
    a name: CPU2
    # number: 2
    # value: 3.1
  {} 2:
    a name: CPU3
    # number: 3
    # value: 5.1
  {} 3:
    a name: CPU4
    # number: 4
    # value: 1.3
[] cpus_filtered:
  {} 0:
    a name: CPU2
    # number: 2
    # value: 3.1
  {} 1:
    a name: CPU3
    # number: 3
    # value: 5.1
```

Code Function example: JSON Filter

Reduce

The `reduce` method allows you to summarize data across an array, with a returned accumulator value. In the example below, a new field, `cpus_reduce`, will be created with a value of `11.8`.

```
__e['cpus_reduce'] = __e['cpus'].reduce((accumulator, entry) => accumulator + entry
```



```
1
2021-06-10
11:31:02.776
-05:00

# _time: 1623342662.776
a arch: Intel x64
[] cpus:
  {} 0:
    a name: CPU1
    # number: 1
    # value: 2.3
  {} 1:
    a name: CPU2
    # number: 2
    # value: 3.1
  {} 2:
    a name: CPU3
    # number: 3
    # value: 5.1
  {} 3:
    a name: CPU4
    # number: 4
    # value: 1.3
# cpus_reduce: 11.8
```

Code Function example: Reduce

Some/Every

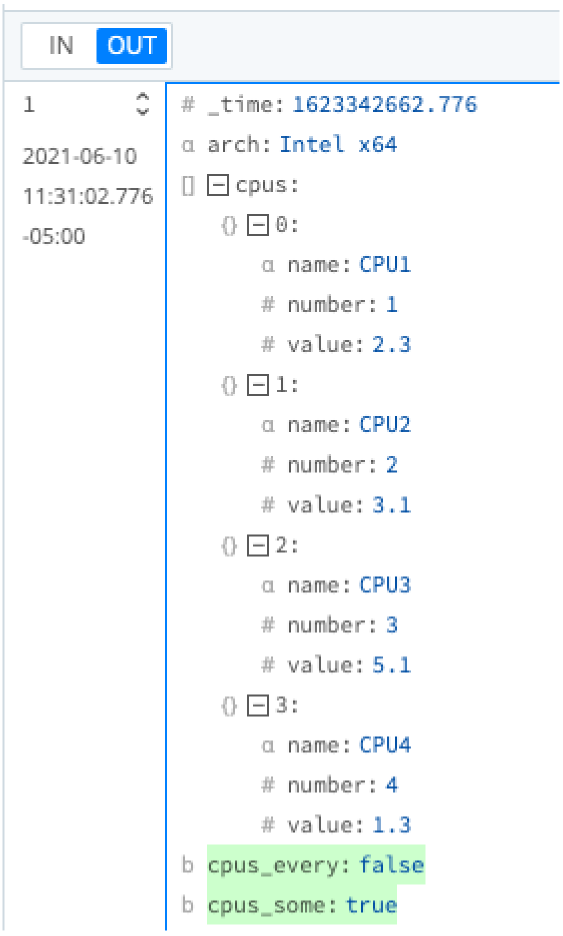
The `some` and `every` methods return a boolean result (`true/false`).

The `some` method checks that there is at least one logic validation that returns `true`. In the example below, `cpus_some` would be set to `true` because there is at least one object with a value greater than or equal to `3`.

```
__e['cpus_some'] = __e['cpus'].some(entry => entry.value >= 3)
```

The `every` method checks that every entry in the array returns `true`. If so, the result is `true`; otherwise, this returns `false`. In the example below, the value for `cpus_every` would be `false`, because not all values in the event are greater than or equal to `10`.

```
__e['cpus_every'] = __e['cpus'].every(entry => entry.value >= 10)
```



```
IN  OUT
1
2021-06-10
11:31:02.776
-05:00
# _time: 1623342662.776
α arch: Intel x64
[] ⊟ cpus:
  {} ⊟ 0:
    α name: CPU1
    # number: 1
    # value: 2.3
  {} ⊟ 1:
    α name: CPU2
    # number: 2
    # value: 3.1
  {} ⊟ 2:
    α name: CPU3
    # number: 3
    # value: 5.1
  {} ⊟ 3:
    α name: CPU4
    # number: 4
    # value: 1.3
b cpus_every: false
b cpus_some: true
```

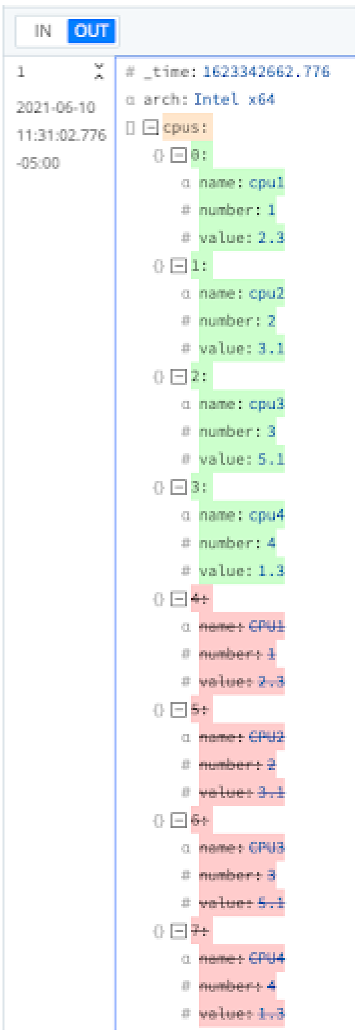
Code Function example: Some/Every

Advanced Examples

Transform a Specific Field

In this example, each `cpus` member will have its `name` field transformed to lowercase. Object.assign is used to keep the original object, while assigning the `name` field to the desired value.

```
__e['cpus'] = __e['cpus'].map(entry => Object.assign(entry, {'name': entry.name.tol
```


IN OUT

1

2021-06-10
11:31:02.776
-05:00

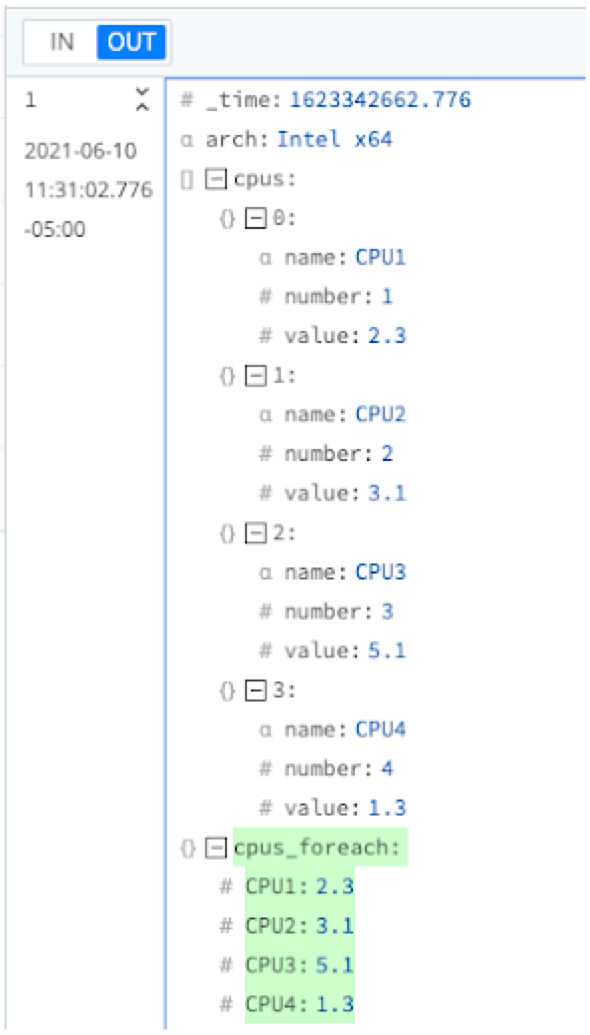
_time: 1623342662.776
arch: Intel x64
cpus:
0:
name: cpu1
number: 1
value: 2.3
1:
name: cpu2
number: 2
value: 3.1
2:
name: cpu3
number: 3
value: 5.1
3:
name: cpu4
number: 4
value: 1.3
~~4:~~
~~name: CPU1~~
~~number: 1~~
~~value: 2.3~~
~~5:~~
~~name: CPU2~~
~~number: 2~~
~~value: 3.1~~
~~6:~~
~~name: CPU3~~
~~number: 3~~
~~value: 5.1~~
~~7:~~
~~name: CPU4~~
~~number: 4~~
~~value: 1.3~~

Code Function example: Transform a Field

forEach

The `forEach` method loops over each element of an array. However, unlike the `map` method, it does not return the value, and it requires a separate temporary variable for result collection.

```
let test = {};
__e['cpus'].forEach((entry, index) => test[entry.name] = entry.value);
__e['cpus_foreach'] = test;
```



```
1

2021-06-10
11:31:02.776
-05:00

# _time: 1623342662.776
a arch: Intel x64
[] ⊟ cpus:
  {} ⊟ 0:
      a name: CPU1
      # number: 1
      # value: 2.3
  {} ⊟ 1:
      a name: CPU2
      # number: 2
      # value: 3.1
  {} ⊟ 2:
      a name: CPU3
      # number: 3
      # value: 5.1
  {} ⊟ 3:
      a name: CPU4
      # number: 4
      # value: 1.3
{} ⊟ cpus_foreach:
  # CPU1: 2.3
  # CPU2: 3.1
  # CPU3: 5.1
  # CPU4: 1.3
```

Code Function example: forEach

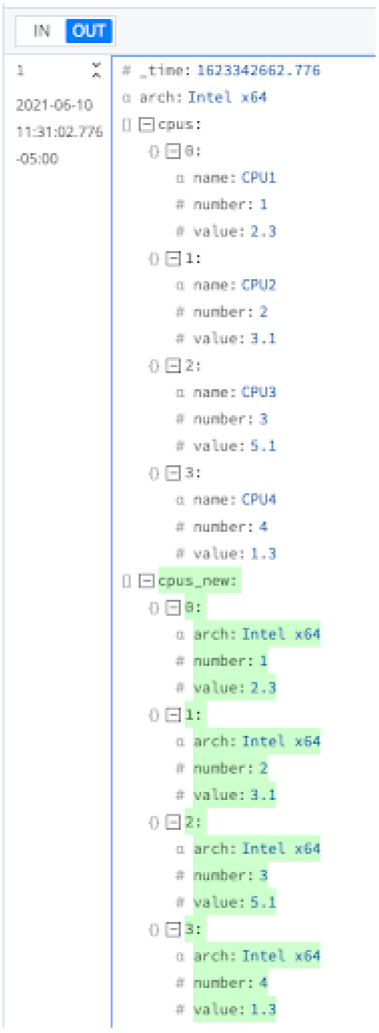
You could also accomplish the same result by replacing the middle line above with the `map` method below:

```
__e['cpus'].map(item => test[item.name] = item.value);
```

Building a New Array

In this example, we create a new array to include some of the original values from each object in the `cpus` array, but we also dynamically inject a new field containing the `arch` field (CPU architecture) from the original event's top level.

```
__e['cpus_new'] = __e['cpus'].map(entry => {
    const container = {};

   for (const [key, value] of Object.entries(entry)) {
        container[key] = value;
    }

   container['arch'] = __e['arch'];

   delete container.name;
    return container;
})
```



IN OUT

```
1
2021-06-10
11:31:02.776
-05:00
```

```
# _time: 1623342662.776
α arch: Intel x64
[] cpus:
  {} 0:
     α name: CPU1
     # number: 1
     # value: 2.3
  {} 1:
     α name: CPU2
     # number: 2
     # value: 3.1
  {} 2:
     α name: CPU3
     # number: 3
     # value: 5.1
  {} 3:
     α name: CPU4
     # number: 4
     # value: 1.3
[] cpus_new:
  {} 0:
     α arch: Intel x64
     # number: 1
     # value: 2.3
  {} 1:
     α arch: Intel x64
     # number: 2
     # value: 3.1
  {} 2:
     α arch: Intel x64
     # number: 3
     # value: 5.1
  {} 3:
     α arch: Intel x64
     # number: 4
     # value: 1.3
```

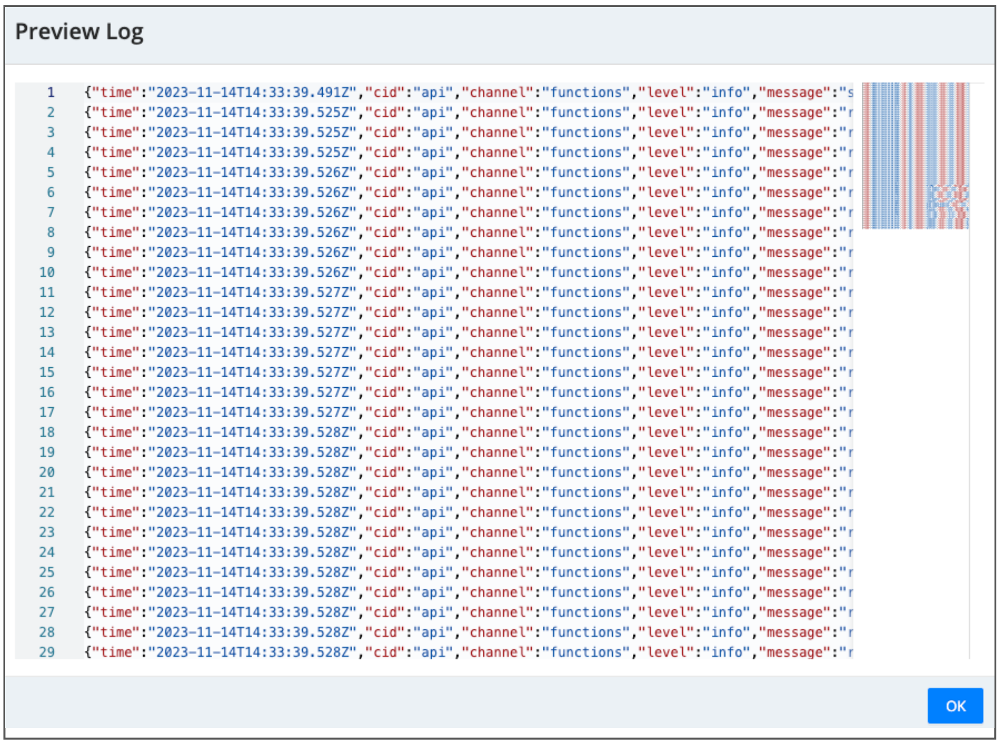
Code Function example: Build a new array

Managing and Troubleshooting Code Execution

Cribl Stream provides the following options for tuning the logic you build with the Code Function.

Logging

Users can access the log messages generated by their Function from the **Preview Log**.



Preview Log

```
{"time":"2023-11-14T14:33:39.491Z","cid":"api","channel":"functions","level":"info","message":"s
{"time":"2023-11-14T14:33:39.525Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.525Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.525Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.526Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.527Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
{"time":"2023-11-14T14:33:39.528Z","cid":"api","channel":"functions","level":"info","message":"r
```

Log messages from a Code Function

Debugging

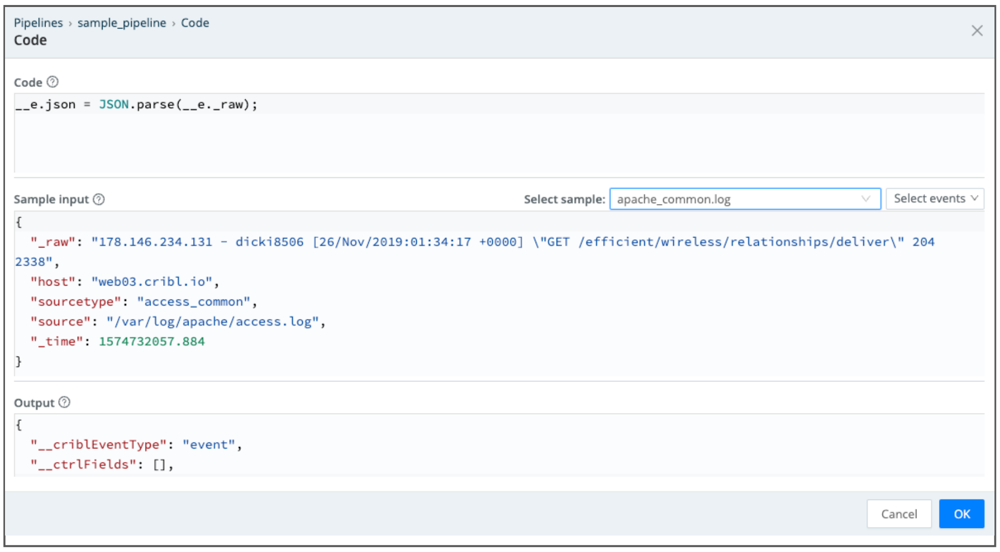
The Code Function's execution context defines a helper function called `debug` that can be used for debugging purposes.

Messages logged by this `debug` helper function are shown in the Preview Log by default.

You can also route these messages to regular logs, although this requires setting the Function's log level (`func:code`) to `debug`.

Previewing

By using the expanded editor, users can run the expression against a sample event, and can preview the transformation:


Pipelines › sample_pipeline › Code
Code
Code
Sample input
Select sample: apache_common.log
Select events
Output
Cancel
OK

```
__e.json = JSON.parse(__e._raw);
```

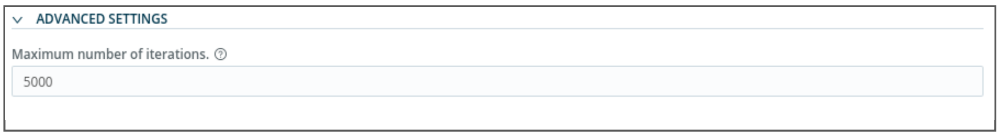
```
{
  "_raw": "178.146.234.131 - dicki8506 [26/Nov/2019:01:34:17 +0000] \"GET /efficient/wireless/relationships/deliver\" 204
2338",
  "host": "web03.cribl.io",
  "sourcetype": "access_common",
  "source": "/var/log/apache/access.log",
  "_time": 1574732057.884
}
```

```
{
  "__criblEventType": "event",
  "__ctrlFields": [],
```

Previewing a Code Function's transformation of an event

Infinite-Loop Protection

The Code Function keeps watching user-defined functions to detect infinite loops that would cause processing to hang. To limit the number of iterations allowed per instance of your Code Function, adjust the **Advanced Settings** > **Maximum number of iterations** option. This defaults to `5,000`; the maximum number allowed is `10,000`. Once the limit is reached, the Code Function will stop processing whatever is after the statement that exhausted the allowed maximum.


ADVANCED SETTINGS
Maximum number of iterations. 5000

Infinite-Loop protection in a Code Function

Loops

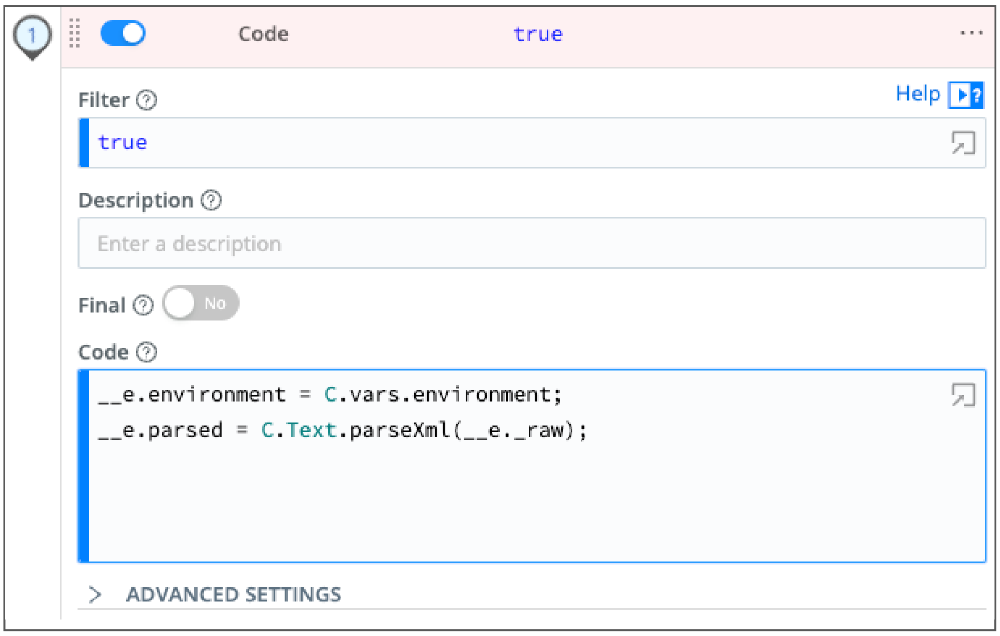
All JavaScript loops and statements are allowed: `for`, `for-of`, `while`, `do-while`, `switch`, etc.

Functions

Users can define their own functions to better organize their code. Both traditional and arrow functions are allowed.

Access to `C.*` Global Object

From the Code's Function body, you can access Cribl Stream's global `C.*` object and its methods/expressions.



Filter: `true`

Code:

```
__e.environment = C.vars.environment;
__e.parsed = C.Text.parseXml(__e._raw);
```

The global `C.*` object of a Code Function

20.4.2. Ingest-Time Fields

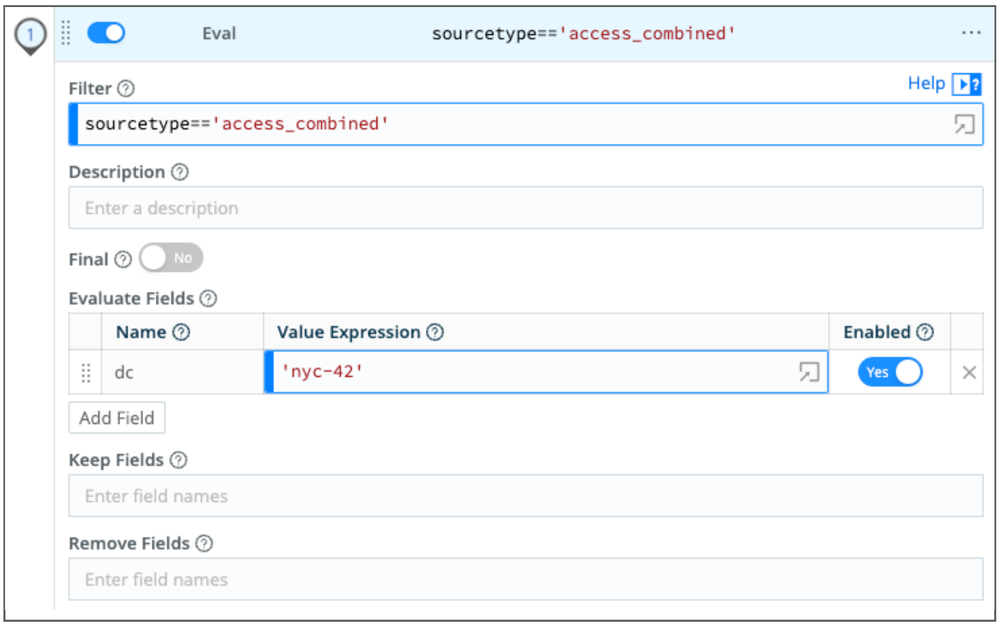
Adding Fields to Data in Motion

To add new fields to any event, we use the out-of-the-box **Eval** Function. We can either apply a Filter to select the events, or we can use the default `true` Filter expression to apply the Function to all incoming events.

Adding Fields Example

Let's see how we add `dc::nyc-42` to all events with `sourcetype=='access_combined'`:

- First make sure you have a Route and Pipeline configured to match desired events.
- Next, let's add an **Eval** function to it with `sourcetype=='access_combined'` as Filter.
- Next, let's click on **Add Field**, add our `dc` field, and click **Save**.



Adding the `dc` field

To confirm, verify that this search returns results: `sourcetype="access_combined" dc::nyc-42`

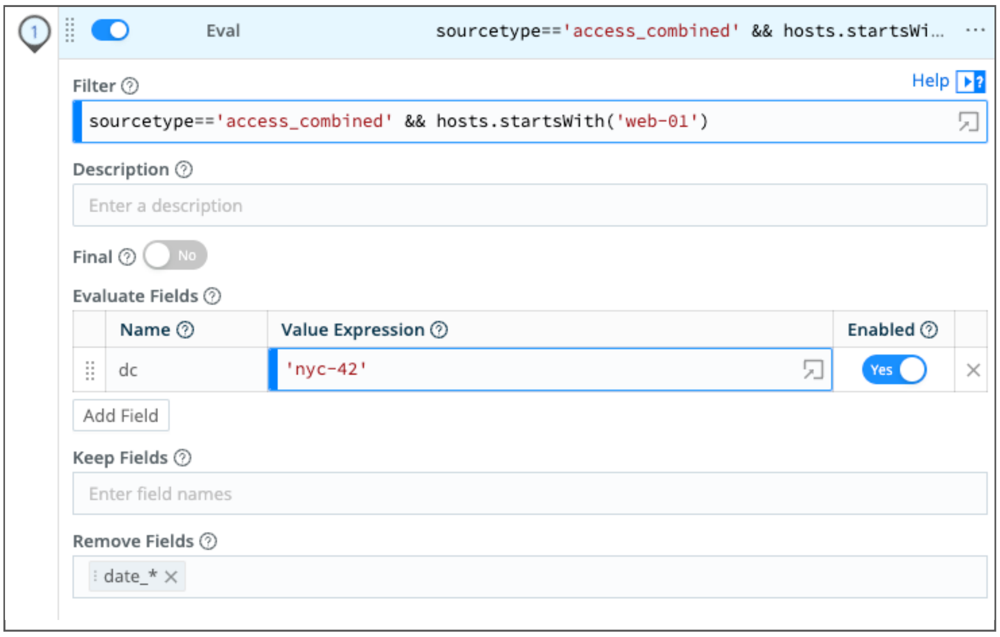
- You can add more conditions to the filter, if you'd like. For example, to limit the field to only events from hosts that start with `web-01`, we can change the filter input to `sourcetype=='access_combined' && hosts.startsWith('web-01')`.

This is a **very** powerful method to change incoming events in real time. In addition to providing the right context at the right time, users can further benefit substantially by using `tstats` for **faster** analytics.

Removing Fields

You can remove fields by listing and/or wildcarding field names. Let's see how we can remove all fields that start with `date_`.:

- First, make sure you have a Route and Pipeline configured to match desired events.
- Next, let's add an **Eval** function to it (as above).
- Next, in **Remove Fields**, add `date_*` and hit Save.



Filter
`sourcetype=='access_combined' && hosts.startsWith('web-01')`

Description
Enter a description

Final: No

Evaluate Fields

Name	Value Expression	Enabled
dc	'nyc-42'	Yes

Add Field

Keep Fields
Enter field names

Remove Fields
date_*

Goodbye `date_` field

To confirm, verify that this search: `sourcetype="access_combined" date_minute=*` will soon stop returning results. Enjoy a more efficient Splunk!

20.4.3. Masking And Obfuscation

Masking and Anonymization of Data in Motion

To mask patterns in real time, we use the out-of-the-box Mask Function . This is similar to `sed`, but with much more powerful functionality.

Masking Capabilities

The Mask Function accepts multiple replacement rules, and accepts multiple fields to apply them to.

Match Regex is a JS regex pattern that describes the content to be replaced. It can optionally contain matching groups. By default, it will stop after the first match, but using `/g` will make the Function replace all matches.

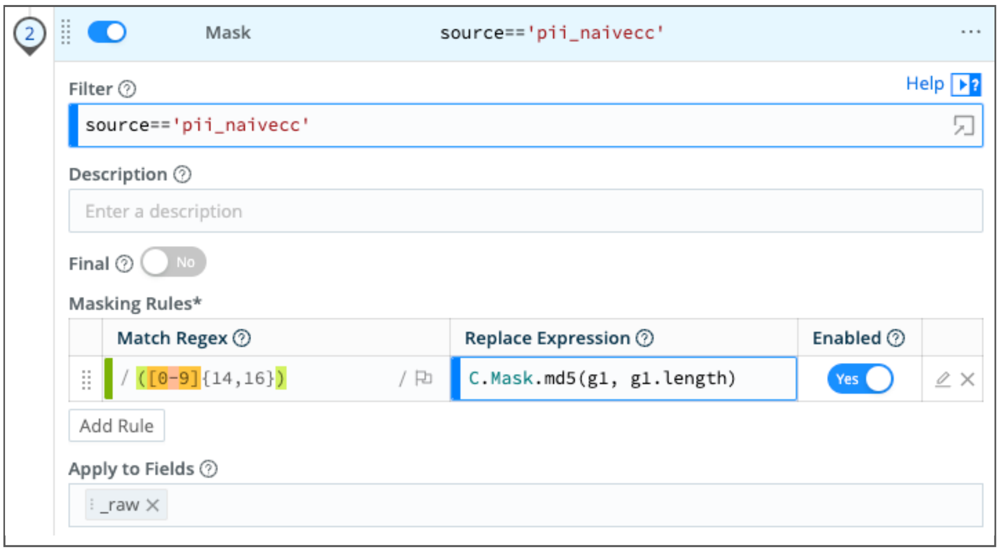
Replace Expression is a JS expression or literal to replace matched content.

Matching groups can be referenced in the **Replace Expression** as `g1`, `g2`… `gN`, and the entire match as `g0`.

There are several masking methods that are available under `C.Mask.`:

`C.Mask.random`: Generates a random alphanumeric string `C.Mask.repeat`: Generates a repeating char/string pattern, e.g., `XXXX` `C.Mask.REDACTED`: The literal 'REDACTED' `C.Mask.md5`: Generates a MD5 hash of given value `C.Mask.sha1`: Generates a SHA1 hash of given value `C.Mask.sha256`: Generates a SHA256 hash of given value

Almost all methods have an optional `len` parameter which can be used to control the length of the replacement. `len` can be either a number or string. If it's a string, its length will be used. For example:


Mask source=='pii_naivecc'

Filter: source=='pii_naivecc'

Description: Enter a description

Final: No

Masking Rules*

Match Regex	Replace Expression	Enabled
/ ([0-9]{14,16}) /	C.Mask.md5(g1, g1.length)	Yes

Add Rule

Apply to Fields: _raw

Defining the replacement length

Masking Examples

Let's look at the various ways that we can mask a string like this one: `cardNumber=214992458870391`. The **Regex Match** we'll use is: `/(cardNumber=)(\d+)/g`. In this example:

- `g0 = cardNumber=214992458870391`
- `g1 = cardNumber=`
- `g2 = 214992458870391`

Random Masking with default character length (4):

- **Replace Expression**: `` `${g1}${C.Mask.random()}` ``
- **Result**: `cardNumber=HRhc`

Random Masking with defined character length:

- **Replace Expression**: `` `${g1}${C.Mask.random(7)}` ``
- **Result**: `cardNumber=neNSm8r`

Random Masking with length preserving replacement:

- **Replace Expression**: `` `${g1}${C.Mask.random(g2)}` ``
- **Result**: `cardNumber=DroJ73qmyaro51u3`

Repeat Masking with default character length (4):

- **Replace Expression**: `` `${g1}${C.Mask.repeat()}` ``
- **Result**: **Result**: `cardNumber=XXXX`

Repeat Masking with defined character choice and length:

- **Replace Expression**: `` `${g1}${C.Mask.repeat(6, 'Y')}` ``
- **Result**: `cardNumber=YYYYYY`

Repeat Masking with length preserving replacement:

- **Replace Expression**: `` `${g1}${C.Mask.repeat(g2)}` ``
- **Result**: `cardNumber=XXXXXXXXXXXXXXX`

Literal REDACTED masking:

- **Replace Expression**: `` `${g1}${C.Mask.REDACTED}` ``
- **Result**: `cardNumber=REDACTED`

Hash Masking (applies to: md5, sha1 and sha256):

- **Replace Expression**: `` `${g1}${C.Mask.md5(g2)}` ``
- **Result**: `cardNumber=f5952ec7e6da54579e6d76feb7b0d01f`

Hash Masking with left N-length* substring (applies to: md5, sha1 and sha256):

- **Replace Expression**: `` `${g1}${C.Mask.md5(g2, 12)}` ``
- **Result**: `cardNumber=d65a3ddb2749` *Replacement length will **not** exceed that of the hash algorithm output; MD5: 32 chars, SHA1: 40 chars, SHA256: 64 chars.

Hash Masking with right N-length* substring (applies to: md5, sha1 and sha256):

- **Replace Expression**: `` `${g1}${C.Mask.md5(g2, -12)}` ``
- **Result**: `cardNumber= 933bfcebf992` *Replacement length will **not** exceed that of the hash algorithm output; MD5: 32 chars, SHA1: 40 chars, SHA256: 64 chars.

Hash Masking with length* preserving replacement (applies to: md5, sha1 and sha256):

- **Replace Expression**: `` `${g1}${C.Mask.md5(g2, g2)}` ``
- **Result**: `cardNumber= d65a3ddb27493f5` *Replacement length will **not** exceed that of the hash algorithm output; MD5: 32 chars, SHA1: 40 chars, SHA256: 64 chars.

20.4.4. Reducing Windows XML Events

Here, we demonstrate how to use just a few Cribl Stream Functions to parse WindowsXML events and reduce their volume by 34–70%, dramatically reducing your downstream infrastructure requirements.

Using Eval and C.Text.parseWinEvent

Cribl Stream's internal C.Text.parseWinEvent method parses Windows XML strings and returns a prettified JSON object. You can use this function within an Eval Function to parse an event or an ad hoc XML string. It works like C.Text.parseXml, but with Windows events, it produces more-compact output.

As you can see from its signature, `C.Text.parseWinEvent` accepts an optional `nonValues` parameter that can further reduce an event's size by discarding characters that you specify as redundant.

(method) `C.Text.parseWinEvent(xml: string, nonValues?: string[]): any`

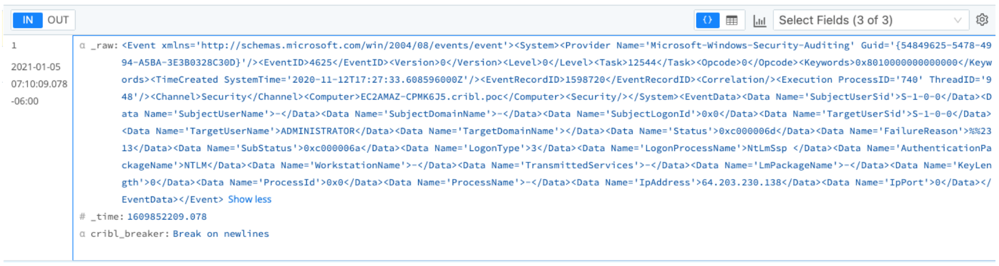
@param – `xml` – an XML string; or an event field containing the XML.

@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.

XML: Threat or Menace?

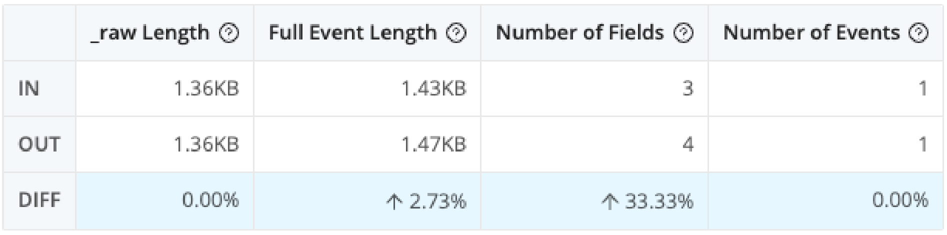
When working with XML, an anonymous Reddit user's quote sums up the challenge: "Some languages can be read by humans, but not by machines, while others can be read by machines but not by humans. XML solves this problem by being readable to neither." An example of a Windows XML event only reinforces this quote:



```
1
2021-01-05
07:10:09.078
-06:00
```

```
α _raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Security-Auditing' Guid='{54849625-5478-49
        94-A5BA-3E3B0328C30D}'/><EventID>4625</EventID><Version>0</Version><Level>0</Level><Task>12544</Task><Opcode>0</Opcode><Keywords>0x8010000000000000</Keyw
        ords><TimeCreated SystemTime='2020-11-12T17:27:33.608596000Z'/><EventRecordID>1598720</EventRecordID><Correlation/><Execution ProcessID='740' ThreadID='9
        48'/><Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></System><EventData><Data Name='SubjectUserSid'>S-1-0-0</Data><D
        ata Name='SubjectUserName'>-</Data><Data Name='SubjectDomainName'>-</Data><Data Name='SubjectLogonId'>0x0</Data><Data Name='TargetUserSid'>S-1-0-0</Data>
        <Data Name='TargetUserName'>ADMINISTRATOR</Data><Data Name='TargetDomainName'></Data><Data Name='Status'>0xc000006d</Data><Data Name='FailureReason'>%%23
        13</Data><Data Name='SubStatus'>0xc000006a</Data><Data Name='LogonType'>3</Data><Data Name='LogonProcessName'>NtLmSsp </Data><Data Name='AuthenticationPa
        ckageName'>NTLM</Data><Data Name='WorkstationName'>-</Data><Data Name='TransmittedServices'>-</Data><Data Name='LmPackageName'>-</Data><Data Name='KeyLen
        gth'>0</Data><Data Name='ProcessId'>0x0</Data><Data Name='ProcessName'>-</Data><Data Name='IpAddress'>64.203.230.138</Data><Data Name='IpPort'>0</Data></
        EventData></Event> Show less
# _time: 1609852209.078
α cribl_breaker: Break on newlines
```

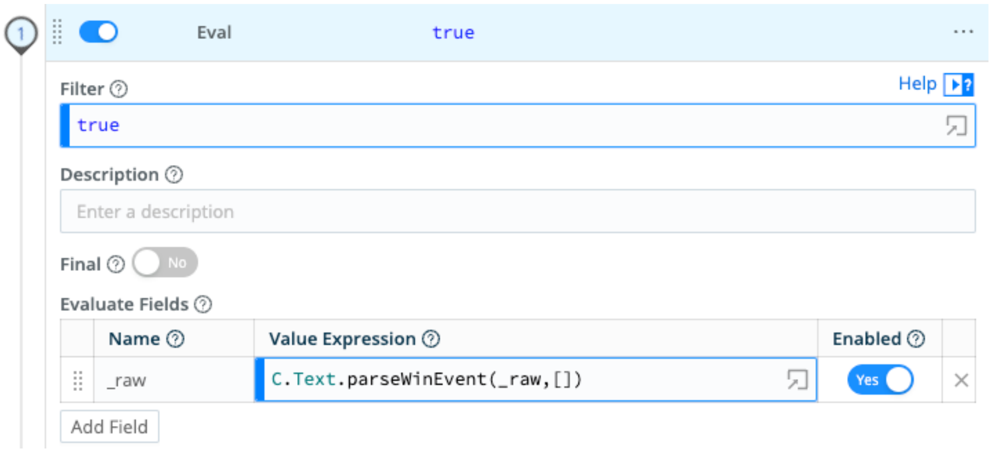
This Windows XML `_raw` event is 1.36KB in size:



	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	1.36KB	1.43KB	3	1
OUT	1.36KB	1.47KB	4	1
DIFF	0.00%	↑ 2.73%	↑ 33.33%	0.00%

Eval to the Rescue

In our initial Eval Function below, the **Value Expression** uses `C.Text.parseWinEvent` to simply parse the `_raw` Windows XML event and turn it into a prettified JSON object:



Eval true

Filter: true

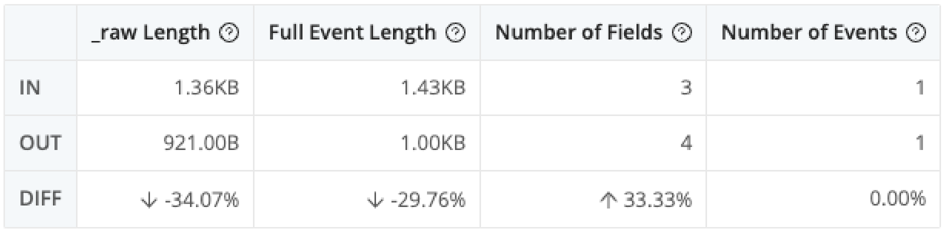
Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression	Enabled
_raw	`C.Text.parseWinEvent(_raw,[])`	Yes

The resulting JSON event is now down to 921.00B in size, a 34.07% reduction of the event:

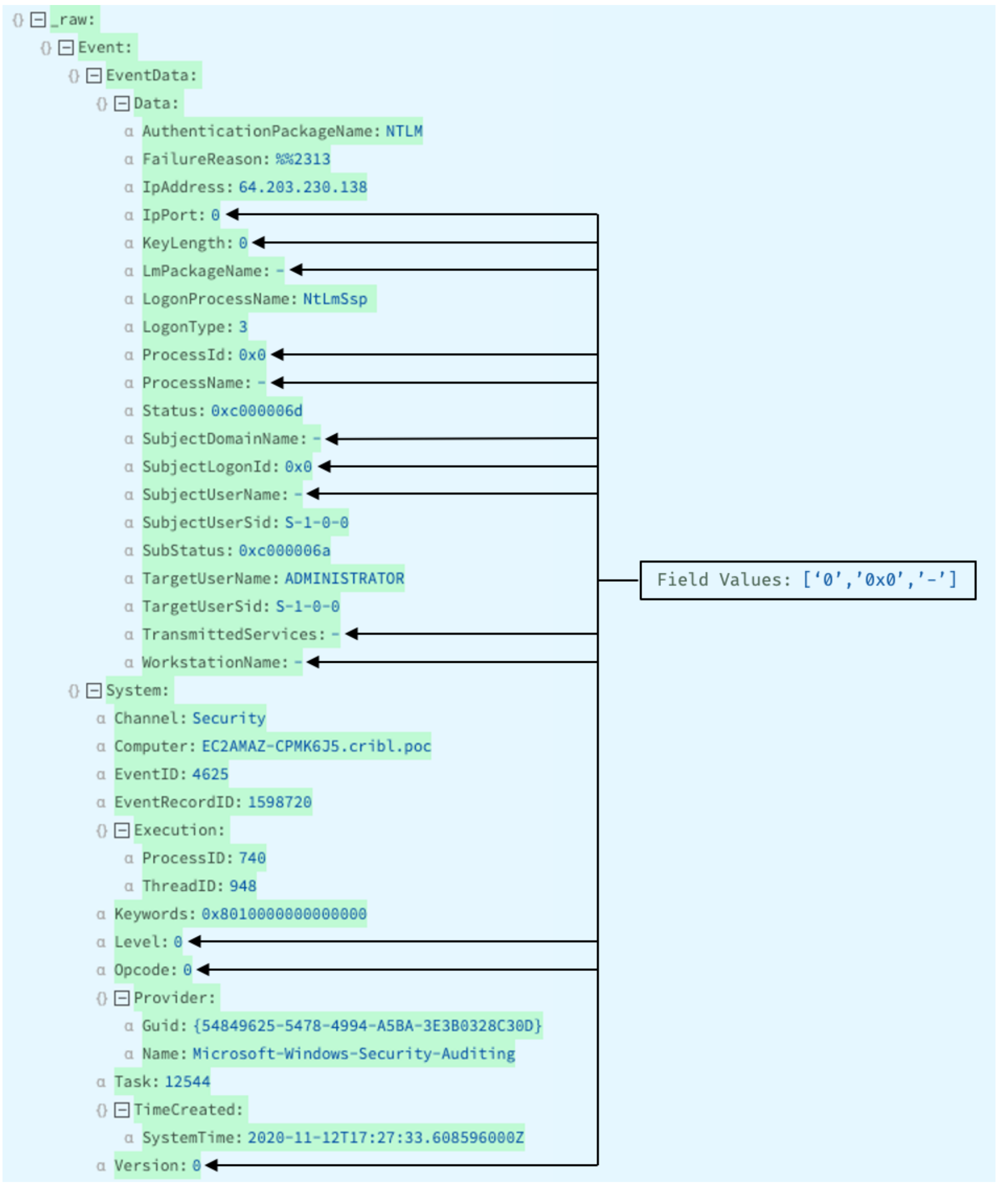


	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	1.36KB	1.43KB	3	1
OUT	921.00B	1.00KB	4	1
DIFF	↓ -34.07%	↓ -29.76%	↑ 33.33%	0.00%

The `C.Text.parseWinEvent(_raw,[])` call yields verbose output that includes XML attributes. If you want flatter output, you can substitute `C.Text.parseXml(_raw, false)`.

Removing Unnecessary Fields with a Better Eval

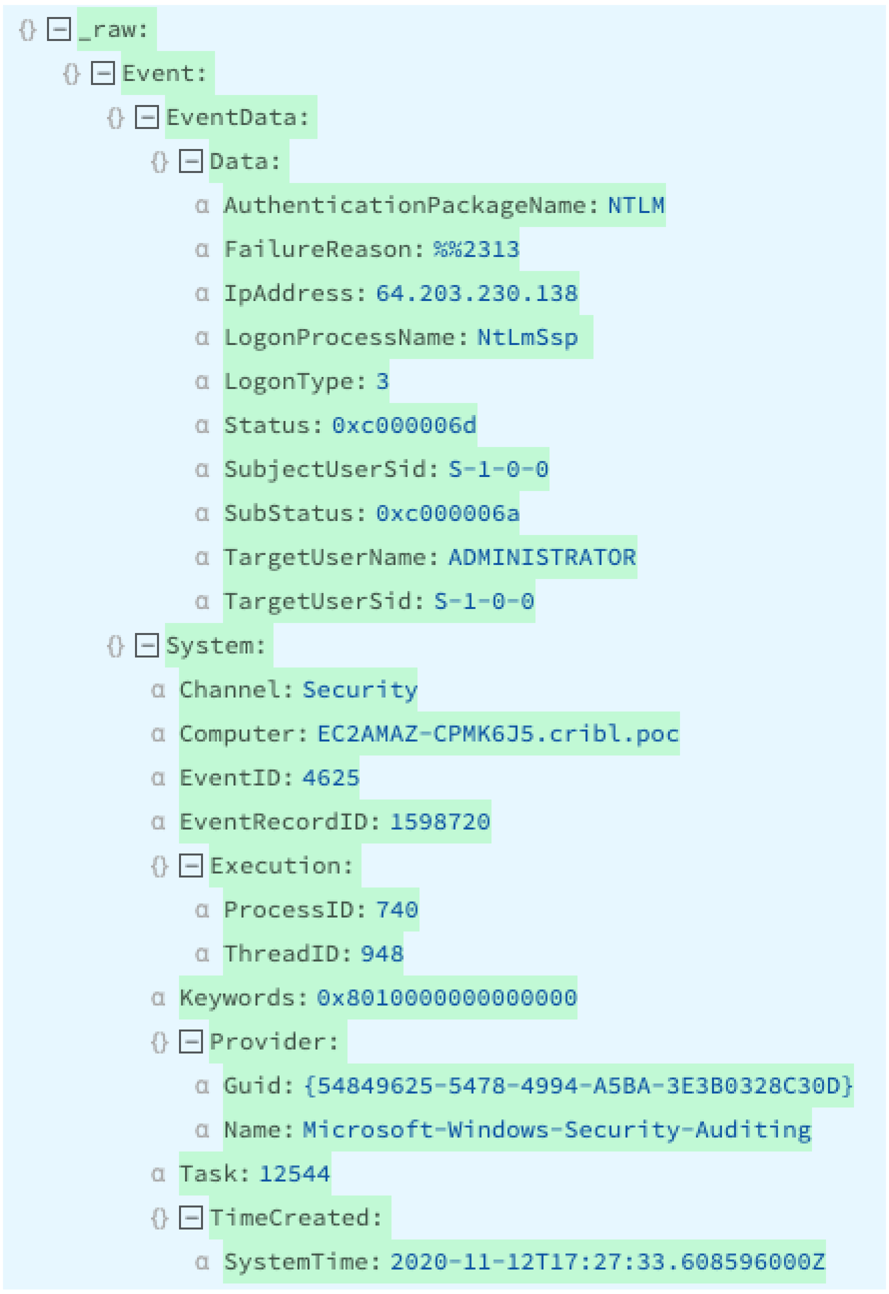
But we can do better. The fields containing essentially null values (`'0'`,`'0x0'`, or `'-'`) bloat events, demanding extra infrastructure and storage:



```
{} ⊟ _raw:
  {} ⊟ Event:
    {} ⊟ EventData:
      {} ⊟ Data:
        α AuthenticationPackageName: NTLM
        α FailureReason: %%2313
        α IpAddress: 64.203.230.138
        α IpPort: 0
        α KeyLength: 0
        α LmPackageName: -
        α LogonProcessName: NtLmSsp
        α LogonType: 3
        α ProcessId: 0x0
        α ProcessName: -
        α Status: 0xc000006d
        α SubjectDomainName: -
        α SubjectLogonId: 0x0
        α SubjectUserName: -
        α SubjectUserSid: S-1-0-0
        α SubStatus: 0xc000006a
        α TargetUserName: ADMINISTRATOR
        α TargetUserSid: S-1-0-0
        α TransmittedServices: -
        α WorkstationName: -
    {} ⊟ System:
      α Channel: Security
      α Computer: EC2AMAZ-CPMK6J5.cribl.poc
      α EventID: 4625
      α EventRecordID: 1598720
      {} ⊟ Execution:
        α ProcessID: 740
        α ThreadID: 948
      α Keywords: 0x8010000000000000
      α Level: 0
      α Opcode: 0
      {} ⊟ Provider:
        α Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
        α Name: Microsoft-Windows-Security-Auditing
      α Task: 12544
      {} ⊟ TimeCreated:
        α SystemTime: 2020-11-12T17:27:33.608596000Z
      α Version: 0
```

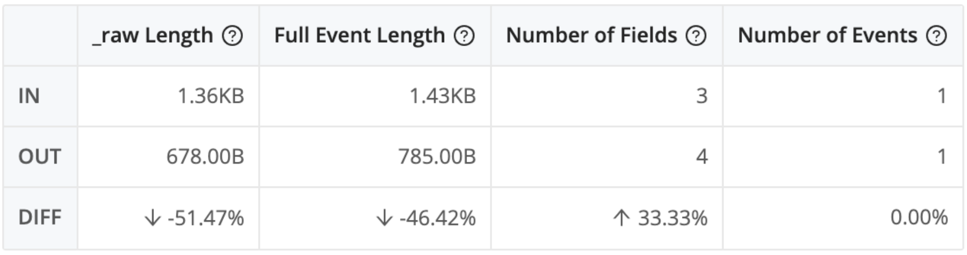
Let's amplify the reduction by removing all of the fields whose values are in the set: `['0','0x0','-']`. This improved version of our Eval Function parses the Windows XML event, and discards `['0','0x0','-']` values. (Its preceding row also tidies up events by removing tabs and curly braces, and replacing newlines and returns with commas.) The result is an even **smaller** prettified JSON object.

If you compare this Preview-pane screenshot to the Preview screenshot above, you can confirm that the fields with values matching `['0','0x0','-']` are removed:



```
{} ⊟ _raw:
    {} ⊟ Event:
        {} ⊟ EventData:
            {} ⊟ Data:
                α AuthenticationPackageName: NTLM
                α FailureReason: %%2313
                α IpAddress: 64.203.230.138
                α LogonProcessName: NtLmSsp
                α LogonType: 3
                α Status: 0xc000006d
                α SubjectUserSid: S-1-0-0
                α SubStatus: 0xc000006a
                α TargetUserName: ADMINISTRATOR
                α TargetUserSid: S-1-0-0
        {} ⊟ System:
            α Channel: Security
            α Computer: EC2AMAZ-CPMK6J5.cribl.poc
            α EventID: 4625
            α EventRecordID: 1598720
            {} ⊟ Execution:
                α ProcessID: 740
                α ThreadID: 948
            α Keywords: 0x8010000000000000
            {} ⊟ Provider:
                α Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
                α Name: Microsoft-Windows-Security-Auditing
            α Task: 12544
            {} ⊟ TimeCreated:
                α SystemTime: 2020-11-12T17:27:33.608596000Z
```

The event is now down to 678.00B in size, translating to a 51.47% reduction from the original event:

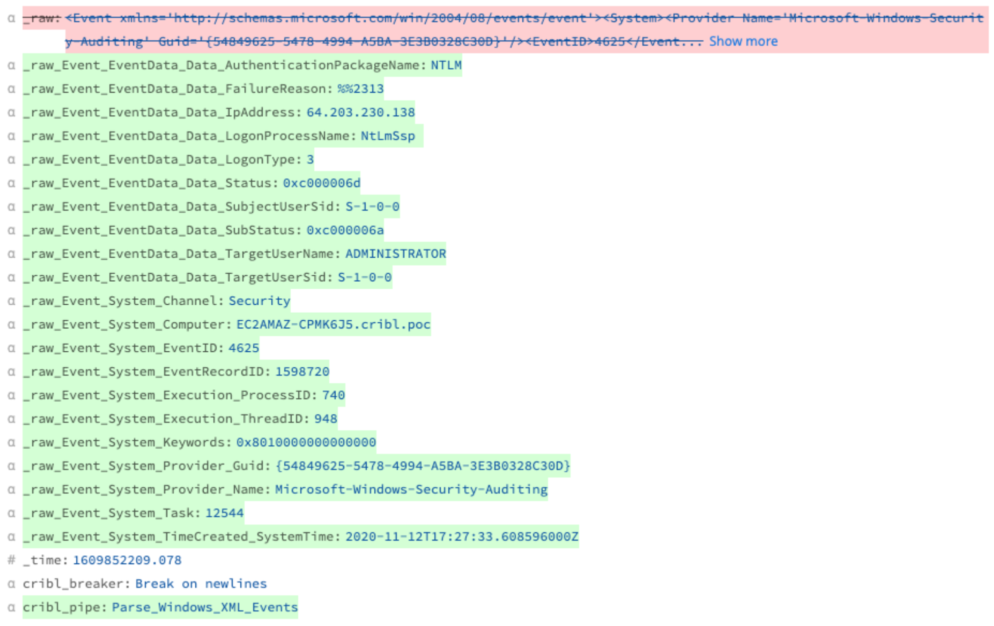


	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	1.36KB	1.43KB	3	1
OUT	678.00B	785.00B	4	1
DIFF	↓ -51.47%	↓ -46.42%	↑ 33.33%	0.00%

Flatten Function

Cribl Stream's Flatten Function is designed to flatten fields out of a nested structure. Let's flatten the JSON object within `_raw`, to see if we can further reduce the event's size before we send it to our preferred destinations.

Using Flatten, we've successfully created top-level fields from the nested JSON structure:



```
α ~~_raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Securit
      y-Auditing' Guid='{54849625-5478-4994-A5BA-3E3B0328C30D}'/><EventID>4625</Event...~~ Show more
α _raw_Event_EventData_Data_AuthenticationPackageName: NTLM
α _raw_Event_EventData_Data_FailureReason: %%2313
α _raw_Event_EventData_Data_IpAddress: 64.203.230.138
α _raw_Event_EventData_Data_LogonProcessName: NtLmSsp
α _raw_Event_EventData_Data_LogonType: 3
α _raw_Event_EventData_Data_Status: 0xc000006d
α _raw_Event_EventData_Data_SubjectUserSid: S-1-0-0
α _raw_Event_EventData_Data_SubStatus: 0xc000006a
α _raw_Event_EventData_Data_TargetUserName: ADMINISTRATOR
α _raw_Event_EventData_Data_TargetUserSid: S-1-0-0
α _raw_Event_System_Channel: Security
α _raw_Event_System_Computer: EC2AMAZ-CPMK6J5.cribl.poc
α _raw_Event_System_EventID: 4625
α _raw_Event_System_EventRecordID: 1598720
α _raw_Event_System_Execution_ProcessID: 740
α _raw_Event_System_Execution_ThreadID: 948
α _raw_Event_System_Keywords: 0x8010000000000000
α _raw_Event_System_Provider_Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
α _raw_Event_System_Provider_Name: Microsoft-Windows-Security-Auditing
α _raw_Event_System_Task: 12544
α _raw_Event_System_TimeCreated_SystemTime: 2020-11-12T17:27:33.608596000Z
# _time: 1609852209.078
α cribl_breaker: Break on newlines
α cribl_pipe: Parse_Windows_XML_Events
```

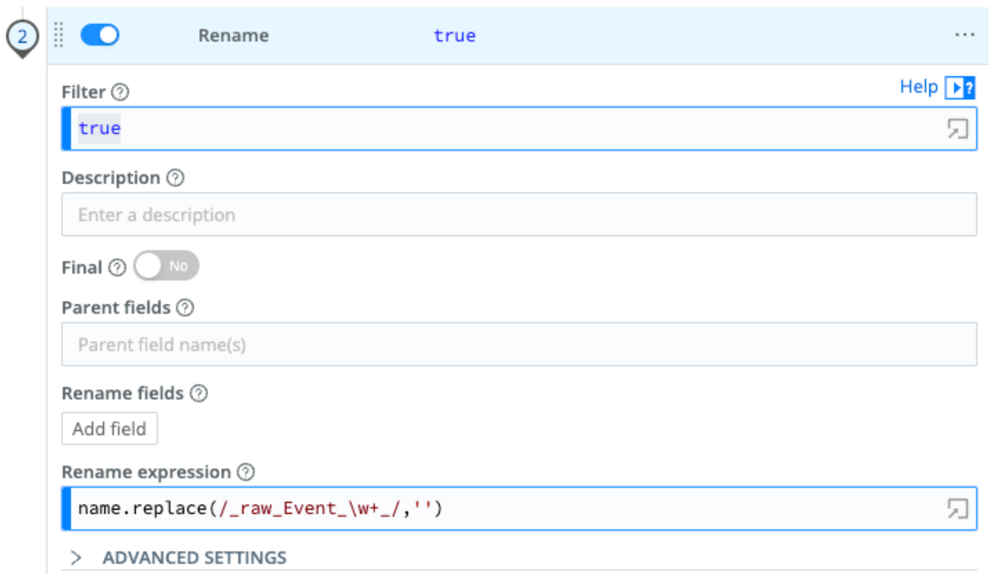
Don't worry about the `_raw` field's deletion (red strikeout). This is the Flatten Function's default behavior. We'll restore `_raw` after we clean and reduce the event even more.

The flattened field names are extracted from `_raw` and delimited with `_`. These field names are quite long. We can optimize them using the Rename Function.

Rename Function

Rename is designed to change fields' names, or to reformat their names (e.g., to normalize names to camelcase). You can use Rename to change specified fields (much like Eval), or to accomplish bulk renaming based on a JavaScript expression (much like the Parser Function). But Rename offers a streamlined way to alter only field names, without other effects.

Let's use Rename to remove any unnecessary prefixes from the field names, to further shrink our events. In the **Rename Expression**, we build a JavaScript expression to match the field names' prefixes (up to the underscore):



Filter
```
true
```
Description: Enter a description

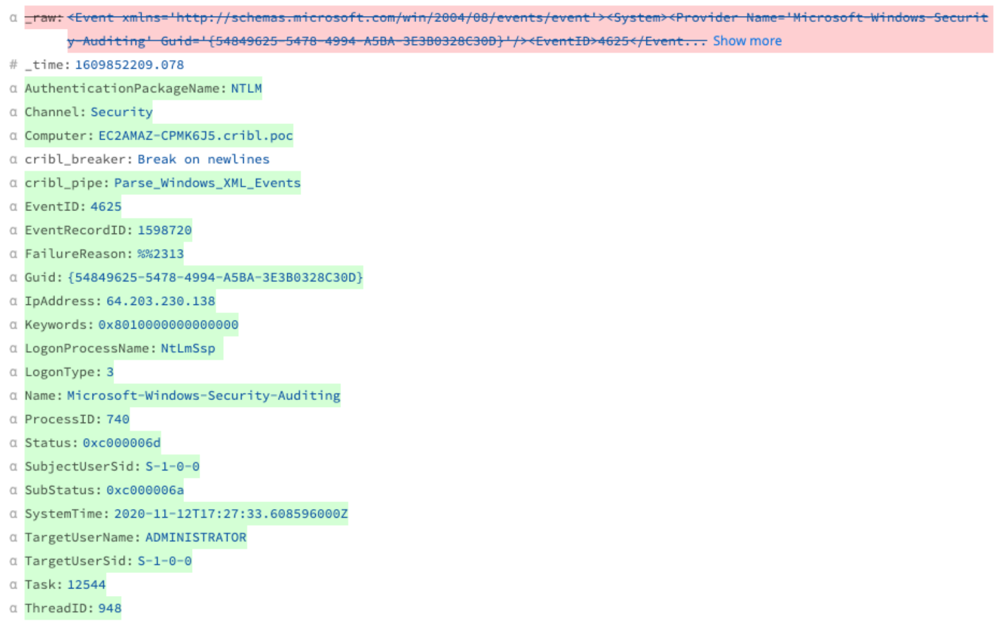
Final: No

Parent fields: Parent field name(s)

Rename fields: Add field

Rename expression
```
name.replace(/_raw_Event_\w+_/,'')
```
ADVANCED SETTINGS

The resulting field names are now much more compact, and easier to work with and manage:



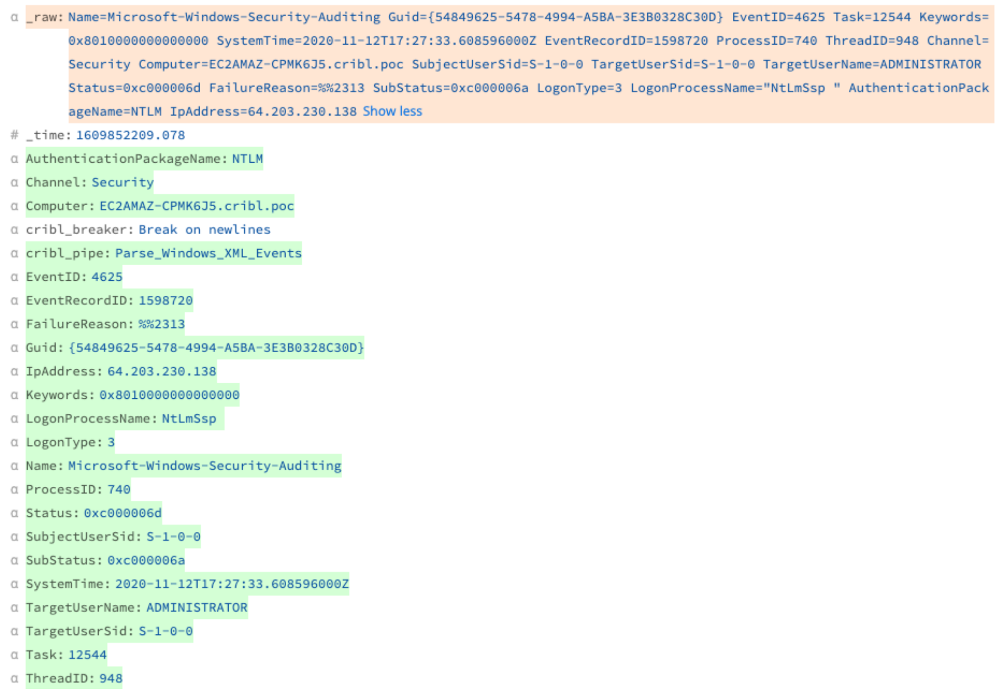
```
α ~~_raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Securit
      y-Auditing' Guid='{54849625-5478-4994-A5BA-3E3B0328C30D}'/><EventID>4625</Event...~~ Show more
# _time: 1609852209.078
α AuthenticationPackageName: NTLM
α Channel: Security
α Computer: EC2AMAZ-CPMK6J5.cribl.poc
α cribl_breaker: Break on newlines
α cribl_pipe: Parse_Windows_XML_Events
α EventID: 4625
α EventRecordID: 1598720
α FailureReason: %%2313
α Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
α IpAddress: 64.203.230.138
α Keywords: 0x8010000000000000
α LogonProcessName: NtLmSsp
α LogonType: 3
α Name: Microsoft-Windows-Security-Auditing
α ProcessID: 740
α Status: 0xc000006d
α SubjectUserSid: S-1-0-0
α SubStatus: 0xc000006a
α SystemTime: 2020-11-12T17:27:33.608596000Z
α TargetUserName: ADMINISTRATOR
α TargetUserSid: S-1-0-0
α Task: 12544
α ThreadID: 948
```

Serialize Function

We started with bloated Windows XML data, and we've parsed and prettified it into JSON. Next, we'll extract key-value pairs. We'll use the Serialize Function, which serializes an event's content into a predefined format.

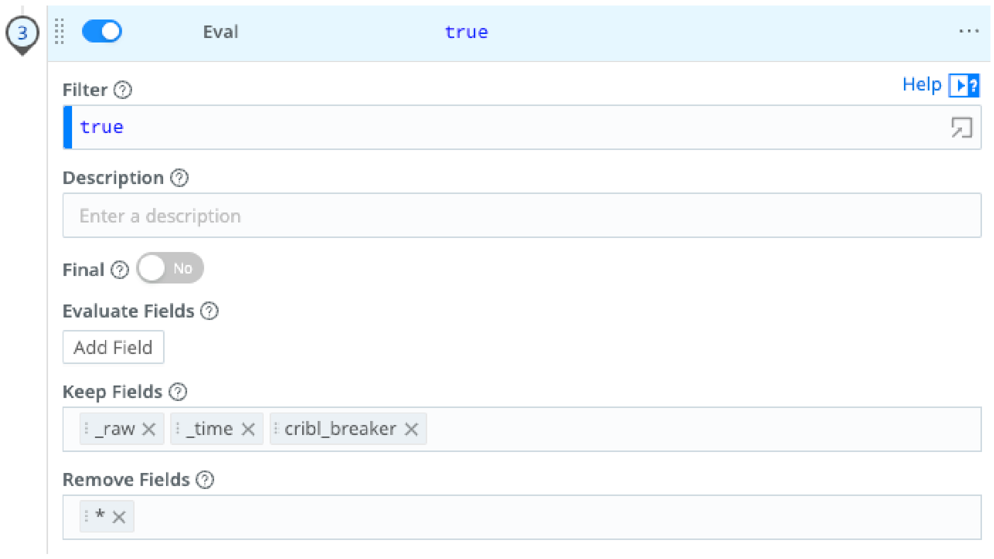
We set Serialize to change the **Type** to key-value pairs. (The Function's other supported target Types include JSON Object and CSV.) Serialize takes the extracted fields and puts them back into `_raw`.

In the Preview pane, the `_raw` field is now back, serialized into compact, tidy key-value pairs:



```
α _raw: Name=Microsoft-Windows-Security-Auditing Guid={54849625-5478-4994-A5BA-3E3B0328C30D} EventID=4625 Task=12544 Keywords=
        0x8010000000000000 SystemTime=2020-11-12T17:27:33.608596000Z EventRecordID=1598720 ProcessID=740 ThreadID=948 Channel=
        Security Computer=EC2AMAZ-CPMK6J5.cribl.poc SubjectUserSid=S-1-0-0 TargetUserSid=S-1-0-0 TargetUserName=ADMINISTRATOR
        Status=0xc000006d FailureReason=%%2313 SubStatus=0xc000006a LogonType=3 LogonProcessName="NtLmSsp " AuthenticationPack
        ageName=NTLM IpAddress=64.203.230.138 Show less
# _time: 1609852209.078
α AuthenticationPackageName: NTLM
α Channel: Security
α Computer: EC2AMAZ-CPMK6J5.cribl.poc
α cribl_breaker: Break on newlines
α cribl_pipe: Parse_Windows_XML_Events
α EventID: 4625
α EventRecordID: 1598720
α FailureReason: %%2313
α Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
α IpAddress: 64.203.230.138
α Keywords: 0x8010000000000000
α LogonProcessName: NtLmSsp
α LogonType: 3
α Name: Microsoft-Windows-Security-Auditing
α ProcessID: 740
α Status: 0xc000006d
α SubjectUserSid: S-1-0-0
α SubStatus: 0xc000006a
α SystemTime: 2020-11-12T17:27:33.608596000Z
α TargetUserName: ADMINISTRATOR
α TargetUserSid: S-1-0-0
α Task: 12544
α ThreadID: 948
```

The last step is to remove any of the extracted fields you don't need before sending events to your destinations. We'll again call on the Eval Function, which adds or removes fields in events. (For a Splunk destination, these are index-time fields.) This final Eval Function looks like this:



3 Eval true

Filter: true

Description: Enter a description

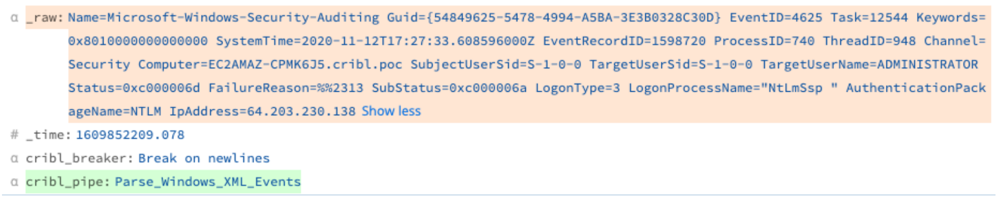
Final: No

Evaluate Fields: Add Field

Keep Fields: _raw, _time, cribl_breaker

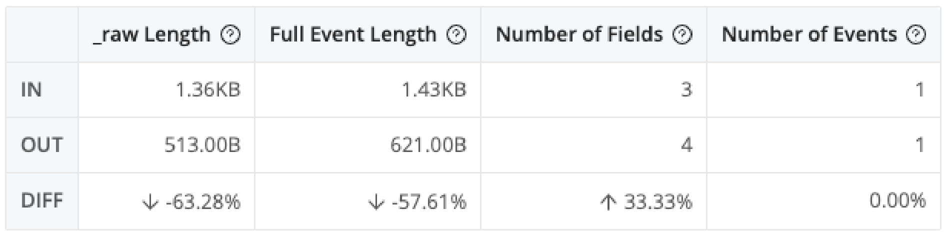
Remove Fields: *

To sum up, we've successfully transformed the original Windows XML event into key-value pairs:



```
α _raw: Name=Microsoft-Windows-Security-Auditing Guid={54849625-5478-4994-A5BA-3E3B0328C30D} EventID=4625 Task=12544 Keywords=
        0x8010000000000000 SystemTime=2020-11-12T17:27:33.608596000Z EventRecordID=1598720 ProcessID=740 ThreadID=948 Channel=
        Security Computer=EC2AMAZ-CPMK6J5.cribl.poc SubjectUserSid=S-1-0-0 TargetUserSid=S-1-0-0 TargetUserName=ADMINISTRATOR
        Status=0xc000006d FailureReason=%%2313 SubStatus=0xc000006a LogonType=3 LogonProcessName="NtLmSsp " AuthenticationPack
        ageName=NTLM IpAddress=64.203.230.138 Show less
# _time: 1609852209.078
α cribl_breaker: Break on newlines
α cribl_pipe: Parse_Windows_XML_Events
```

And we've dramatically reduced the event's size, while retaining all of the necessary fields. The event is now down to 513.00B in size, translating to a 63.28% reduction from the original Windows XML:



	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	1.36KB	1.43KB	3	1
OUT	513.00B	621.00B	4	1
DIFF	↓ -63.28%	↓ -57.61%	↑ 33.33%	0.00%

Try This at Home

Below is an export of the whole Cribl Stream Pipeline presented here. Import this JSON to experiment with it and modify it to match your own needs:

Win XML Pipeline

```
{
  "id": "Windows_Security_Events",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "filter": "true",
        "conf": {
          "comment": "This Cribl Stream Pipeline reduces Microsoft Windows XML eve
        },
        "id": "comment"
      },
      {
        "filter": "true",
        "conf": {
          "add": [
            {
              "name": "_raw",
              "value": "_raw.replace(/[{}\\t]/gm,'').replace(/[\\n\\r]+/gm,',')"
            },
            {
              "name": "_raw",
              "value": "C.Text.parseWinEvent(_raw,['0x0','0','-'])"
            }
          ]
        },
        "id": "eval",
        "disabled": false,
        "description": "Remove tabs & curly braces; replace newlines & returns wit
      },
      {
        "filter": "true",
        "conf": {
          "fields": [
            "_raw"
          ],
          "prefix": "",
          "depth": 5,
          "delimiter": "_"
        },
        "id": "flatten",
        "disabled": false,
        "description": "Flatten the object into key value fields"
      },
      {
        "filter": "true",
        "conf": {
          "baseFields": [],
          "renameExpr": "name.replace(/_raw_Event_\\w+_/,'')",
          "rename": []
        },
        "id": "rename",
        "disabled": false,
        "description": "Rename the top level fields and remove the log suffix"
      },
      {
        "filter": "true",
        "conf": {
          "type": "kvp",
          "fields": [
            "!_*",
            "!cribl_*",
            "!index",
            "!host",
            "!source",
            "!sourcetype",
            "*"
          ],
```

Sample data

```
      },
      "id": "serialize",
      "disabled": false,
      "description": "Serialize top level events back into _raw in the desired ty
    },
    {
      "filter": "true",
      "conf": {
        "keep": [
          "_time",
          "_raw"
        ],
        "remove": [
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Prov
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Prov
                        SeBackupPrivilege
                        SeRestorePrivilege
                        SeTakeOwnershipPrivilege
                        SeDebugPrivilege
                        SeSystemEnvironmentPrivilege
                        SeLoadDriverPrivilege
                        SeImpersonatePrivilege
                        SeDelegateSessionUserImpersonatePrivilege
                        SeEnableDelegationPrivilege</Data></EventData></Event>
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Prov
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Prov
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Prov
```

20.4.5. Regex Filtering

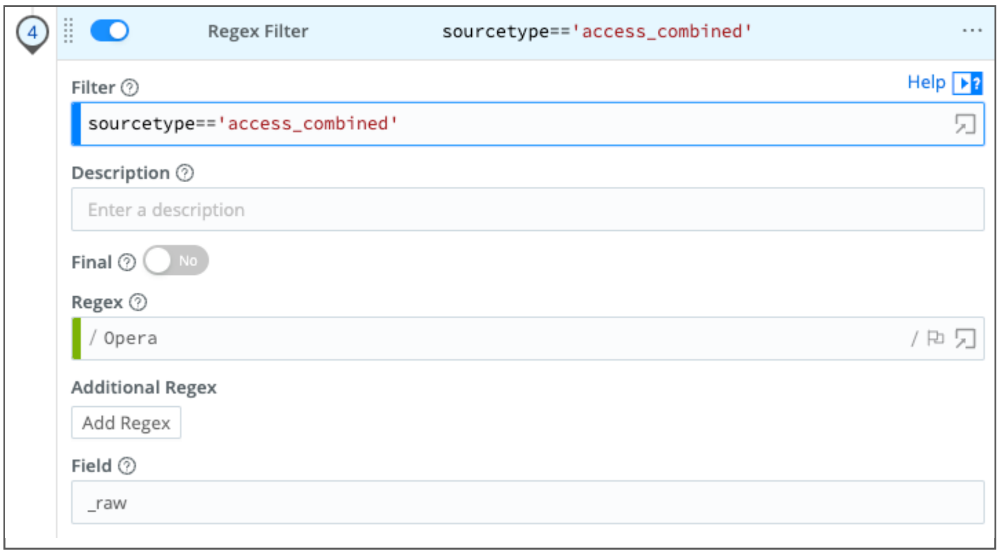
To filter events in real time (data in motion), we use the out-of-the-box **Regex Filter** Function. This is similar to nullqueueing with TRANSFORMS in Splunk, but the matching condition is way more flexible.

Regex Filtering Example

Let's see how we can filter out any `sourcetype=='access_combined'` events whose `_raw` field contains the pattern `Opera`:

First, make sure you have a Route and Pipeline configured to match desired events.

Next, let's add a **Regex Filter** Function to it:



sourcetype=='access_combined'

Filter: sourcetype=='access_combined'

Description: Enter a description

Final: No

Regex: / Opera

Additional Regex: Add Regex

Field: _raw

Defining the Regex Filter Function

Next, verify that this search does **not** return any results: `sourcetype="access_combined" Opera`

You can add more conditions to the Filter input field. For example, to further limit the filtering to only events from hosts with domain `dnto.ca`, change the filter input to `sourcetype=='access_combined' && host.endsWith('dnto.ca')`.

This is a very flexible method for filtering incoming events in real time, on virtually any arbitrary conditions.

21. Troubleshooting

21.1. Known Issues

This page lists known issues affecting Cribl Stream and/or Cribl Edge.

2024-03-20 v.4.5.1 – Permissions error appears when you try to edit a Pipeline [CRIBL-23627]

Problem: Cribl Stream Project Members with the Editor permission cannot view or edit Pipelines within the Stream Project.

Workaround: None.

Fix: In Cribl Stream 4.6.

2024-03-18 – v4.5.1 – An error can occur when you click Save to create a new Worker Group or Fleet [CRIBL-23496]

Problem: In Edge and Stream, an error can occur when you click **Save** to create a new Worker Group or Fleet. This error appears as a warning: `The Config Helper service is not available because a configuration file doesn't exist or the settings are invalid. Please fix it and restart Cribl server.`

Workaround: Refresh the page.

Fix: In Cribl Stream 4.6.

2024-03-14 – v4.5.1 – Amazon S3 Source and Destination, Amazon SQS Destination, and CrowdStrike Source crash in FIPS mode [CRIBL-23436]

Problem: When Cribl Stream in FIPS mode attempts to read or write files, the Amazon S3 Source and Destination, Amazon SQS Destination, and CrowdStrike Source crash with a `digital envelope routines::unsupported` error. This happens because the underlying Amazon AWS SDK v2 is configured to use the MD5 algorithm, which is not allowed in FIPS mode. Cribl Stream 4.6.0 fixes this problem by configuring the AWS SDK to skip the checksum computation that used MD5.

Workaround: None.

Fix: In Cribl Stream 4.6.

2024-03-13 – v4.4.0-v.4.5.1 – Diagnostic bundles cannot be created when git is enabled [CRIBL-23374]

Problem: Attempting to create a diagnostic bundle fails when git is enabled, resulting in an error.

Workaround: In the **Create & Export Diag Bundle** window, disable **Include git**.

Fix: In Cribl Stream 4.6.

2024-03-12 v.4.4.4-v.4.5.0 – S3 Collector Path extractors setting has no effect [CRIBL-23338]

Problem: In the S3 Collector, the **Collector Settings** > **Path extractors** setting has no effect because of a defect in the `index.js` file.

Workaround: Contact Cribl Support for assistance replacing the `index.js` file with a corrected one.

Fix: In Cribl Stream 4.5.1.

2024-02-13 – v4.5.1 – Edge Leaders running 4.5.1 can’t use the Legacy upgrade method to upgrade 4.5.0 Nodes when jobs/tasks are disabled [CRIBL-22801]

Problem: Edge Leaders on version 4.5.1 that have Legacy upgrades enabled and **Enable Jobs/Tasks** toggled to `No` cannot upgrade their 4.5.0 Edge Nodes. The job finishes with the task timing out, and the Edge Node is not upgraded.

Workaround: On the 4.5.1 Leader, enable Legacy upgrades **and** toggle **Enable Jobs/Tasks** to `Yes`. Commit and deploy as necessary. The Leader will be able to create a job with the upgrade task and upgrade its Edge Nodes to 4.5.1.

Fix: Version TBD.

2024-02-12 – v4.5.0 – The API Reference automatic authorization doesn’t work in Cribl.Cloud [CRIBL-22822]

Problem: In Cribl.Cloud, the API Reference (under **Global Settings**) uses an invalid access token. Because of this, you will receive an HTTP 401 error when you attempt to run an API command.

Workaround: To obtain a valid token, do the following:

1. Open the Developer Tools network tab in your browser and copy the authorization bearer token from one of your requests. Alternatively, you can follow the docs steps here to create a Cribl.Cloud API credential and obtain a token. Copy the token.
2. On the API Reference page, click **Authorize**.
3. In the resulting modal, click **Logout**.
4. Enter your copied token in the **Value** field and click **Authorize**.

Fix: In Cribl Stream 4.5.1.

2024-02-12 – v4.5.0 – When in the context of a Pack, the Pipeline link in the routing table is broken [CRIBL-22762]

Problem: In the Pack context, under **Routes**, if you click the hyperlinked Pipeline name in the table, you will get an `Item not found` error. Clicking `Try again` from the error page does not resolve the issue. However, the link icon at the end of the Pipeline field in the routing table works as expected and brings you to the intended Pipeline page.

Workaround: Use the link icon at the end of the Pipeline field in the Route to access the Pipeline.

Fix: In Cribl Stream 4.5.1.

2024-02-09 – v4.5.0 – OTLP Metrics function does not parse `__criblMetrics` when metric values are strings [CRIBL-22731]

Problem: Documentation for the Publish Metrics Function indicates that it evaluates the **Metric Name Expression** to the **Event Field Name**, but it actually returns a `null`. As a result, the OTLP Metrics function is unable to properly parse `__criblMetrics` when the metrics values are strings.

Workaround: When using Publish Metrics and OTLP Metrics together, use the Numerify Function before the OTLP Metrics Function.

Fix: In Cribl Stream 4.5.1.

2024-02-09 – v4.5.0 – Fleet-level log searches no longer work [CRIBL-22716]

Problem: The **Edge** > **Manage** > **Logs** UI no longer supports searching logs across entire Fleets. The option will be removed in an upcoming release.

Workaround: Search individual nodes or across Fleets in Cribl Search instead.

Fix: In Cribl Stream 4.5.1.

2024-01-30 – v4.1.0 through Current – Product Documentation: Reference Architecture image download link is broken [CRIBL-22502]

Problem: The editable diagram that appears in the All-in-One Reference Architecture section of the documentation is broken for all non-current release versions. Clicking the link for the `SVG` or `draw.io` file download results in a 403 error.

Workaround: Use the links in the most recent version of the documentation.

Fix: Version TBD.

2024-01-19 – v4.4.0-v4.4.4 – Unable to upgrade Leader from 4.3.x to 4.4.x when an external KMS is enabled [CRIBL-22299]

Problem: When attempting to upgrade the Leader from 4.3.x to 4.4.x with an external KMS enabled, you will receive several `Secret failed to decrypt` errors and the web UI will fail to load. This is due to an issue with how `client.secret` is handled when an external KMS is enabled.

Workaround: None.

Fix: In Cribl Stream 4.5.

2024-01-19 – v3.1.0 through Current – Notification targets created in one Cribl product can be deleted from another without warning [CRIBL-22292]

Problem: When you create a Notification target, it will be made available to all products in the Cribl suite. When you attempt to delete a target from the same application you created it in (for example, Cribl Stream), the product will properly prevent the deletion when there is an active notification using the target. However, if you switch to a new application (for example, Cribl Search), you will be allowed to delete the target without error or warning. This will leave the active notification in the original product (Stream) with an undefined target. You should not be able to delete the target if it is used by any active notification in any Cribl suite product.

Workaround: None.

Fix: In Cribl Stream 4.5.1.

2024-01-16 – v4.4.4 – Already-existing Webhook Destinations show unexpected UI behavior upon upgrade to v4.4.4 [CRIBL-22184]

Problem: If you have a Webhook Destination in a Cribl Stream deployment, and then upgrade that deployment to version 4.4.4, the UI will show an error; the new **Load balancing** toggle will be turned on with associated settings displayed; and the UI will not allow you to save changes.

Workaround: First, toggle **Load balancing** off. This will return the UI to normal behavior. Then, if you want to use the Destination without load balancing, configure as usual and save. Or, if you prefer, update the **Webhook URLs** to include one or more valid endpoints you want to send to, and save the Destination with **Load balancing** enabled.

Fix: Version TBD.

2024-01-05 – v4.0.0-v4.4.3 – Windows Event Forwarder Source used up its allowed quota of connections [CRIBL-21965]

Problem: When the Windows Event Forwarder (WEF) Source could not process a request coming from a Windows client, it failed to correctly close the socket or send an error response to the client. Eventually, maximum allowable active connections could be reached, stopping the flow of data.

Workaround: For any affected WEF Source, set **Advanced Settings** > **Max active requests** to `0` to allow unlimited requests. Note that this will eventually cause other problems as the number of open-inactive sockets grows, meaning that the Worker nodes will eventually need to be restarted.

Fix: In Cribl Stream 4.4.4.

2023-12-21 – v.4.0.0-v4.5.1 – Splunk TCP Source logs include an unclear message about an unsupported payload [CRIBL-21845]

Problem: Because the Splunk TCP Source does not support ingesting compressed data via the Splunk S2S protocol, it cannot parse such payloads, and a correct error message in this situation would say “Could not parse payload. Turn compression off in upstream sender and try again.” Instead, the Source logs the unclear message “Failed to parse S2S payload”.

Workaround: None.

Fix: In Cribl Stream 4.6.

2023-12-13 v.4.0.0–4.4.3 – Sources incorrectly report failure [CRIBL-21690]

Problem: When using OpenTelemetry HTTP, the Source always listens on the default address (either `0.0.0.0` or `::`, or both, depending on the network configuration of the host), ignoring the option you configured in **General Settings** > **Address**.

Workaround: Unidentified.

Fix: In Cribl Stream 4.4.4.

2023-11-30 – v4.4.0-v.4.4.1 – Edge-only Sources are unavailable in standalone Edge instances [CRIBL-21443]

Problem: Edge-only Sources are disabled in standalone (not distributed) Edge instances.

Workaround: None.

Fix: In Cribl Stream 4.4.2.

2023-11-21 – v.4.4.0–4.4.3 – Azure Data Explorer Destination exposes irrelevant PQ settings in Cribl.Cloud [CRIBL-21335]

Problem: For Worker Nodes in Cribl.Cloud with Azure Data Explorer (ADX) Destinations: such Destinations' **Persistent Queue Settings** should expose only a **Clear Persistent Queue** button. They are incorrectly also exposing the PQ settings required for on-prem deployments.

Workaround: In Cribl.Cloud ADX Destinations, ignore all Persistent Queue settings except the **Clear Persistent Queue** button.

Fix: In Cribl Stream 4.4.4.

2023-11-16 – v4.4.0–4.4.3 – "Unsupported system page size" error when upgrading the Leader[CRIBL-21289]

Problem: Fedora users can experience an "unsupported system page size" error when upgrading the Leader. This is due to an issue with `jemalloc`.

Workaround: You can manually remove `jemalloc.so`. Docker users can use the following:

```
docker ... -e CRIBL_BEFORE_START_CMD_0="rm /opt/cribl/bin/libjemalloc.so" ...
```

Fix: In Cribl Stream 4.4.4.

2023-11-15 – v4.4.0 through Current – With Leader HA/Failover, creating diag bundles via CLI produces incorrect results [CRIBL-21258, CRIBL-21557]

Problem: For deployments configured for Leader high availability/failover, creating diagnostic bundles via the UI works normally but creating them via the CLI does not. In Cribl Stream v.4.4.0 and newer, the operation fails with a `not a git repository` error. In v.4.4.0 and older, the diag bundle created contains obsolete data. In both cases the underlying cause is that the process that creates the diag bundle cannot access the `CRIBL_CONF_DIR` environment variable.

Workaround: Use the UI to create the diag bundle. Or, if you use the CLI, preface the `diag` command with the needed environment variable, like this: `CRIBL_CONF_DIR=/<path_to_failover_volume> cribl diag create`. If you have an `instance.yml` config file, you can find the path to the failover volume in the `distributed` section.

Fix: Error message improvements were made as part of CRIBL-21258. Fix for CRIBL-21557 is TBD.

2023-11-09 – All Versions through Current – High-volume UDP data dropped in Cribl.Cloud [SAAS-4399]

Problem: Ingesting high rates of UDP events per second can cause Cribl.Cloud to drop some of the data. This limitation of the UDP protocol affects UDP-supporting Sources: Raw UDP, Metrics, SNMP Trap, and Syslog in UDP mode.

Workaround: To minimize the risk of data loss, deploy a hybrid Stream Worker Group, with Worker Nodes as close to the UDP senders as possible.

Cribl also recommends tuning the OS UDP buffer size.

Fix: Version TBD.

2023-11-09 – v.4.4.0–4.4.1 – Go version update required [CRIBL-21155]

Problem: The Go version requires an update.

Workaround: None.

Fix: In Cribl Stream 4.4.2.

2023-11-08 – v.4.4.0–4.4.1 – Azure Data Explorer Destination fails to validate database settings in Streaming mode [CRIBL-21123]

Problem: When Azure Data Explorer Destination is in **Streaming** mode and **Validate database settings** is turned on, database validation will fail unless **Ingestion service URI** has been set in **Batching** mode first.

Workaround: Switch to **Batching** mode, set the **Ingestion service URI** field, and save the Destination. Then switch back to **Streaming** mode and save the Destination again.

Fix: In Cribl Stream 4.4.2.

2023-11-07 – v.4.4.0–4.4.3 – Stream Groups' Members indicators always show zero count [CRIBL-21099, CRIBL-22072]

Problem: On a Stream Worker Group's **Manage** > **Overview** page, the **Members** counters show `0` in all Roles/Permissions, even when Members have been added.

Workaround: Navigate to **Group Settings** > **Members** for an accurate view of all Members and their access levels.

Fix: Fix for CRIBL-22072 in Cribl Stream 4.4.4. (CRIBL-21099 closed as a duplicate.)

2023-11-07 – v.4.4.0–4.4.1 – Worker Node generates an incomplete diagnostic bundle when Include git is enabled [CRIBL-21093]

Problem: When you teleport to a Worker Node and create a diagnostic bundle, if you leave **Include git** toggled on in the **Create & Export Diag Bundle** modal, the bundle that the system creates will be malformed and missing information, and the UI will produce a fatal error notification.

Workaround: When you create diagnostic bundles, turn **Include git** off.

Fix: In Cribl Stream 4.4.2.

2023-11-06 – v.4.4.0–4.4.1 – Azure Data Explorer Destination does not correctly turn off Validate database

settings [CRIBL-21060]

Problem: Once the Azure Data Explorer Destination has been saved with **Validate database settings** on, turning the setting off has no effect, and the Destination performs validation anyway.

Workaround: To ensure that **Validate database settings** is turned off, perform either of these two workarounds:

Option 1: Clone the Destination or create a new one, and turn **Validate database settings** off before saving.

Option 2: Turn off **Validate database settings**, then restart the Cribl Stream Worker Node.

Fix: In Cribl Stream 4.4.2.

2023-10-12 – v.4.3.0–4.4.1 – Rapid logging causes Worker Node memory spikes and prevents logging [CRIBL-20607]

Problem: A large number of logs written in a short time causes memory spikes in Worker Nodes and stops log output.

Workaround: Unidentified.

Fix: In Cribl Stream 4.4.2.

2023-10-05 – v.4.3.0 – Microsoft Windows Installer (MSI) does not create an Edge desktop shortcut [CRIBL-20504]

Problem: When you install Cribl Edge on Windows, the MSI doesn't create the expected desktop shortcut for Edge, even if you select **Create a shortcut for Cribl Edge on the desktop**.

Workaround: None.

Fix: In Cribl Stream 4.3.1.

2023-10-03 – v.4.3.0–4.4.1 – Upgrading to v.4.3.0 or later causes Amazon Kinesis Streams and WEF Sources to lose state [CRIBL-20444]

Problem:

After an upgrade to v.4.3.0, Worker Nodes for a Cribl Stream Amazon Kinesis Streams Source or Windows Event Forwarder (WEF) Source can lose state upon init. This can be especially problematic for customers with large streams and high data ingestion.

An Amazon Kinesis Streams Source that loses state will fail to read a data stream from the point where it most recently left off. Instead, the Source will start reading from the location configured by **Optional Settings** > **Shard iterator start**.

- If **Shard iterator start** is set to `Earliest Record` (the default), the Source can start too far behind in the stream to ever catch up.
- If **Shard iterator start** is set to `Latest Record`, the Source can start later than where it left off, causing some records to be skipped.

A Microsoft Windows Event Forwarder (WEF) Source (with bookmarks configured for the subscription) that loses state will fail to request that upstream clients send events starting at a bookmarked location. Instead, Cribl Stream will send no bookmark, prompting upstream clients to send **all** events that otherwise match the subscription. This can cause Cribl Stream to ingest duplicate events from WEF clients.

Workaround: If you have not yet upgraded to v.4.3.0 or later, consider delaying the upgrade until this issue is fixed.

Fix: In Cribl Stream 4.4.2.

2023-09-15 – v.4.3.0 – Scheduled (cron-based) jobs fail to load for some Sources and Collectors [CRIBL-20088]

Problem: Collection tasks for Office 365 Sources, the Splunk Search Source, the Prometheus Scraper Source, and scheduled Collectors fail to load when the Leader restarts, fails over, or upgrades. Collectors running jobs ad hoc are not affected.

Workaround: If you're using one of the listed Sources or use scheduled Collector jobs on-prem, do not upgrade to 4.3.0. If you are already on 4.3.0, downgrade to 4.2.2, or wait and upgrade to 4.3.1 when it's available.

You can also restart the config helper for on-prem installations using the command:
`pkill -f CONFIG_HELPER`

You must run this command each time the Leader fails over, or is restarted (ideally, not too often).

Fix: In Cribl Stream 4.3.1.

2023-09-14 – v.4.3.0 – Chain Function's link breaks when the chained Pipeline/Pack is resaved [CRIBL-20083]

Problem: After linking a Chain Function to a Pipeline or Pack, resaving the target Pipeline/Pack triggers `Invalid link` errors.

Workaround: 1. Copy the Pipeline containing the Chain Function to your clipboard, delete the original Pipeline, and then paste the copy to resolve the error. 2. Alternatively, on-prem admins can restart the config helper that manages the Worker Group. (In the UI, navigate to **Settings** > **Global** > **System** > **Services** > **Processes**.)

Fix: In Cribl Stream 4.3.1.

2023-09-14 – v.4.3.0 – In Cribl.Cloud, Admin Members cannot modify Global Settings [CRIBL-20038]

Problem: After upgrading to 4.3.0, Members granted the Admin Permission cannot modify Global Settings.

Workaround: Members with the Owner Permission can still modify Global Settings.

Fix: In Cribl Stream 4.3.1.

2023-09-14 – v.4.3.0 – Worker Nodes cannot be upgraded in Leader-managed upgrades [CRIBL-20086]

Problem: In Stream and Edge, if you upgrade the Leader to 4.3.0, then use **Commit & Deploy** to push configs to Worker Groups before upgrading Worker Nodes to 4.3.0, you won't be able to upgrade the Worker Nodes from the Leader thereafter.

This issue affects only Leader-managed upgrades of Worker Nodes.

Workaround: If a Leader is on 4.3.0, you should upgrade all Worker Nodes to 4.3.0 before making any config changes or committing and deploying to the Nodes.

If you have already committed and deployed to Worker Nodes that are on a version prior to 4.3.0 (and the Leader is on 4.3.0), here are three workaround options:

- Revert and redeploy the last commit, then upgrade the Worker Nodes and deploy the most up-to-date changes, **OR**
- Upgrade the Worker Nodes via command line or script, **OR**
- Downgrade the Leader to 4.2.2 and then upgrade to 4.3.1 when it is available.

If you run into issues, contact support@cribl.io for resolution assistance.

Fix: In Cribl Stream 4.3.1.

2023-09-13 – v.4.2.1 through Current – Edge upgrades occasionally fail when Fleets contain a large number of Nodes [CRIBL-20067]

Problem: Leader-managed upgrades occasionally fail when a Fleet contains a large number of Nodes.

Workaround: Restart the upgrade operation to upgrade additional Nodes.

Fix: Version TBD.

2023-09-13 – v.4.3.0 – Any Code Function before another Function breaks the Data Preview OUT tab display [CRIBL-20040]

Problem: Any Code Function inserted before the end of a Pipeline prevents the Data Preview OUT tab from rendering properly. This occurs even if the Code Function contains only a single comment, or is blank. Adding a Code Function at the end of a Pipeline does not cause an issue. This is a Data Preview UI issue only – if you capture events to a Destination, Cribl Stream processes them as expected.

Workaround: None.

Fix: In Cribl Stream 4.3.1.

2023-08-31 – v.4.2.2 – When in XML format, the Windows Event Logs Source doesn't accept log names that contain spaces [CRIBL-19818]

Problem: When the Windows Event Logs Source is set to XML format, any configured log names that contain spaces (e.g., `Windows Powershell`) cause an error and the log can't be read.

Workaround: None.

Fix: In Cribl Edge 4.3.

2023-08-28 – v.4.2.2 through 4.3.0 – PQ doesn't drain for the Event Hubs Destination when Acknowledgements is set to All or Leader [CRIBL-19756]

Problem: When the `Acknowledgements` setting is set to `All` or `Leader`, Kafka-based Destinations (especially Azure Event Hubs) can fail to drain the persistent queue (PQ) and the logs are filled with `Attempting to send faster than the downstream can receive – consider checking the network connection and broker health` errors.

Workaround: You can set a limit using the `Persistent Queue Drain rate limit (EPS)` setting (for example, set it to `500`) to get the PQ to drain when the datagen is turned off. However, if the rate of data input continues to exceed the rate at which the Destination can send events downstream (which is slowed by the `Acknowledgements` setting), the PQ will continue to build. Alternatively, you can set the `Acknowledgements` setting to `None`, which significantly increases the rate at which data can be sent to Event Hubs.

Fix: In Cribl Stream 4.3.1.

2023-08-24 – v.4.2.0 through 4.2.2 – Deleting a Source via QuickConnect doesn't refresh the view [CRIBL-19709]

Problem: When using QuickConnect, if you delete a Source by opening its drawer and clicking `Delete Source`, the Source is deleted. However, it will continue to show on the page until you manually refresh the page.

Workaround: Manually refresh the page after deleting a Source.

Fix: In Cribl Stream 4.3.0.

2023-08-23 - v.4.0.0 through 4.4.4 - Packs installed in Fleets are not visible to Subfleets. [CRIBL-19676]

Problem: When a Fleet has Subfleets, and you install a Pack in the parent Fleet, none of the Subfleet's Pipeline drop-downs make the Pack available. This is true for Routes, pre-processing Pipelines in Sources, and post-processing Pipelines in Destinations.

Workaround: Install the Pack both in the parent Fleet, and in any Subfleets that need to use the Pack.

Fix: In Cribl Stream 4.5.

2023-08-23 – v.4.0–4.2.x – Inherited Packs in Fleets lose their configured Routes [CRIBL-19700]

Problem: Routes in a Pack at the Fleet level revert to default Routes when inherited by a SubFleet.

Workaround: None.

Fix: Couldn't reproduce the error.

2023-08-16 – v.4.2.2 – When adding or editing a Route, typeahead misbehaves in the Filter field [CRIBL-19571]

Problem: Entering text in the **Filter** field of a Route is difficult in Cribl Stream 4.2.2 because the typeahead function behaves erratically, sometimes moving the cursor around or inserting text fragments.

Workaround: 1. Assemble the filter expression in a different Cribl expression field, or in any text editor, then copy and paste it here. 2. Alternatively, click the Manage as JSON button at the **Data Routes** table's upper right. Then, in the JSON editor, build the expression as the `"filter":` key's value.

Fix: In Cribl Stream 4.3.

2023-08-02 - v.4.2.1 through Current - Healthy Destinations spuriously report `Blocked Status` [CRIBL-19322]

Problem: Healthy TCP-based, Kafka-based, Splunk, and Metrics Destinations can spuriously report `Blocked Status` in the **Charts** tab, sometimes also logging the blocked status inaccurately.

Workaround: None.

Fix: Version TBD.

2023-07-26 – v.4.2.0–4.2.1 – File Monitor Source causes memory leaks and Worker 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 Worker 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: In Cribl Stream 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: In Cribl Stream 4.2.2.

2023-07-19 – v.4.2.2 – Group Information for Cribl.Cloud Worker Groups not showing on summary page for SSO users [CRIBL-19107]

Problem: On Cribl.Cloud Organizations, when you open a Stream Group's **Manage** > **Overview** page, the **Group Information** section is not populated for users imported using SSO from external identity providers. (This section populates correctly for users/Members configured natively within your Cribl Organization.)

Workaround: None.

Fix: Couldn't reproduce the error.

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: In Cribl Stream 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: In Cribl Stream 4.2.2.

2023-07-19 – v.4.2.0 – Credential encryption errors preventing user access to Cribl Stream Sources and Destinations [SAAS-4823]

Problem: A critical regression caused Worker Groups to become unresponsive, preventing access to Sources and Destinations. This issue was caused by incorrectly encrypted credentials.

Workaround: Upgrade to Cribl Stream 4.2.1.

Fix: In Cribl Stream 4.2.1.

2023-07-18 – v.4.2.0 through Current – 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.4.2.0–4.4.4 – Product-level Editor Permissions do not allow a commit/deploy action on the Worker Groups/Fleets page [CRIBL-18970, CRIBL-20113]

Problem: Users with product-level Edge **Editor** or Stream **Editor** Permissions are unable to commit and deploy changes made to Worker Groups or Fleets from the Group or Fleet page. However, if the user navigates to the Group or Fleet where the change was saved, the **Commit and Deploy** button is available.

Workaround: Commit and deploy from the Worker or Group where you saved the change.

Fix: In Cribl Stream 4.5. CRIBL-18970 is closed and expanded to CRIBL-20113.

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: In Cribl Stream 4.2.2.

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: In Cribl Stream 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: In Cribl Stream 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: In Cribl Stream 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: In Cribl Stream 4.2.2.

2023-07-16 – v.4.2.0–4.4.3 – 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 Stream 4.1.x will no longer be able to access the **Monitoring** tab and dashboard after upgrading to Cribl Stream 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: In Cribl Stream 4.4.4.

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: In Cribl Stream 4.2.2.

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: Not observed as of Cribl Stream 4.2.2.

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: In Cribl Stream 4.3.

2023-07-12 – 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: In Cribl Stream 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: In Cribl Stream 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: In Cribl Stream 4.3.

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: In Cribl Stream 4.3.

2023-07-03 – v.4.2.0–4.3.0 – 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: In Cribl Stream 4.3.1.

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 Stream 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 Stream displays a spurious error about insufficient permissions.

Workaround: Use the list’s check boxes to select Pipelines, then click **Delete Selected Pipelines**.

Fix: In Cribl Stream 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 Stream 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 through Current – Datadog Agent Source fails to receive metrics [CRIBL-18281]

Problem: When a Datadog Agent using Datadog API v2 sends metrics, Datadog Agent Source fails to receive them, because the Source uses Datadog API v1. Datadog Agent logs will show `API Key invalid, dropping transaction` errors. Debug logging will likely produce `Dropping request because of unallowed path message` errors with `statusCode: 403`, which Datadog Agent interprets as "invalid API key."

Workaround: In the `datadog.yaml` file, ensure that `use_v2_api.series` is set to `false`. Cribl Stream's Datadog Agent Source will then receive metrics normally, since both the Source and the Datadog Agent will be using Datadog API v1, which Datadog still supports.

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 Stream 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: In Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream environments.

Workaround: Roll back to a previous stable version, such as 4.1.0.

Fix: In Cribl Stream 4.1.3.

2023-05-26 v.4.1.2–4.1.3 – Worker Nodes don't update from Cribl Stream Leader [CRIBL-17792]

Problem: In Cribl Stream 4.1.2–4.1.3, if you define the `baseURL` key on a Leader, Worker 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 4.1.3.

2023-05-05 – v.4.1.1 – S3 ingestion slows after upgrade [CRIBL-17317]

Problem: After upgrading to Cribl Stream 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 Stream 4.1.2.

2023-04-27 – v.4.1.1 – Worker Nodes integrated with `systemd/initd` can fail upon reconfig [CRIBL-17264]

Problem: Where 4.1.1 on-prem or hybrid Worker Nodes (on Linux) are integrated with `systemd/initd`, deploying new config bundles to those Worker Nodes can stop the Worker 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 Stream 4.1.2.

2023-04-19 v.4.4.0–4.5.0 – Special characters in Source or Destination auth tokens cause authentication failure [CRIBL-17047]

Problem: When you include special characters in a Source or Destination auth token, those characters may cause an authentication failure.

Workaround: Use only alphanumeric characters or the underscore (`_`) in a Source or Destination auth token. Do not use any of the following characters: `<`, `>`, `"`, `` ` ``, `\r`, `\n`, `\t`, `{`, `}`, `|`, `\`, `^`, or `'`. Alternatively, upgrade to Cribl Stream 4.5.1.

Fix: In Cribl Stream 4.5.1.

2023-04-17 – All Versions through v4.2.2 – Kafka-based Destinations fail to report backpressure properly [CRIBL-16944]

Problem: With a downstream receiver exerting backpressure, Kafka-based Destinations do not report backpressure promptly, even though they effectively exert backpressure. As a result, upstream data might stall for several minutes before appropriate errors surface in the Destination's health status, and PQ (if configured) will not engage promptly.

Workaround: Decreasing the value for the `Request timeout (ms)` setting within the `Advanced Settings` section of the destination configuration might lead to the Destination reporting backpressure more promptly.

Fix: In Cribl Stream 4.2.2.

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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Worker Groups to 4.1.x before you deploy to them.

Fix: In Cribl Stream 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.4.1.0-4.3.1 – 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: In Cribl Stream 4.4.

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 Stream 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: In Cribl Stream 4.3.1. Closed as a duplicate of CRIBL-15467 which was fixed in 4.1.1.

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 Stream 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 Worker Nodes in the same Worker Group, which generates duplicate data.

Workaround: Toggle **Minimize Duplicates** to `No` in **Azure** > **Event Hubs** > **Advanced Settings**.

Fix: In Cribl Stream 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 Stream 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 Stream 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: In Cribl Stream 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 Stream 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 Stream 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 Stream will send an endless series of errors to the log file.

Workaround: Correct the **Environment** value or leave it empty.

Fix: In Cribl Stream 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 Stream 4.1.1.

2023-02-20 – v.4.0.3–4.1.0 – On-prem Worker Nodes automatically upgrade, ignoring Leader settings [CRIBL-15367]

Problem: Upgrading a Leader Node causes its on-prem Worker 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 Cribl.Cloud Worker Nodes.

Workaround: 1. Toggle **Enable automatic upgrades** to `Yes` and save the configuration; then slide it back to `No` and save again. Or: 2. Edit `local/cribl/cribl.yml` to explicitly set `upgradeSettings.disableAutomaticUpgrade` to `true`.

Fix: In Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream Leader's Master URL via `instance.yml` causes Worker Nodes to ignore the `CRIBL_DIST_WORKER_PROXY` env var. Instead of trying to connect to the proxy configured in `CRIBL_DIST_WORKER_PROXY`, Worker Nodes will try to connect to other entities (e.g., Leader Nodes) directly, producing unexpected results.

Workaround: Set the Cribl Stream Leader's Master URL via the `CRIBL_DIST_MASTER_URL` env var, rather than via `instance.yml`.

Fix: In Cribl Stream 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 Stream 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 Stream 4.0.4.

2023-01-25 – v.3.5.x–4.3.1 – 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: In Cribl Stream 4.4.

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 Stream 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 Stream 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 Worker Nodes [CRIBL-14299]

Problem: Modifying a Lookup table on the Leader doesn't always propagate the changes to Worker Nodes, even after clicking **Commit** and **Deploy**.

Workaround: Manually restart the Worker Nodes to refresh Lookup tables.

Fix: In Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 4.0.3.

2022-12-01 – v.4.0.0 – Amazon S3 Source stops receiving data after upgrade [CRIBL-14093]

Problem: Upgrading to Cribl Stream 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 Stream 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 Stream 4.0.3.

2022-11-28 – v.4.1.0 through Current – Diag bundles might fail to download from teleported Worker Nodes [CRIBL-13999]

Problem: When you’re remotely accessing a Worker 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 Worker Node’s UI before creating the diag.

Fix: Version TBD.

2022-11-23 – All versions through 4.1.0 – Worker 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, Worker Nodes sometimes fail to automatically restart with the correct configuration changes.

Workaround: Manually restart the Worker Nodes.

Fix: In Cribl Stream 4.1.1.

2022-11-23 – v.4.0.0 – Teleported Worker 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 Worker 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 Worker Group.

Fix: In Cribl Stream 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 Stream 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 Cribl.Cloud as of 4.0.1. On-prem customers should use the workaround, which will be added to the deployment documentation as a standard practice.

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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream does not perform BPS calculations for internal logs or metrics. But the visual disparity can be confusing.

Fix: In Cribl Stream 4.1.

2022-11-03 – v.4.0.0 – Sorting by column on Manage > Worker Node page results in blank list [CRIBL-13505]

Problem: When you click a column header on the **Manage** > **Worker 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 Stream 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: None, works as designed.

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 Stream 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 Stream 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 Stream 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 Stream 4.1.

2022-09-27 – 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 Stream has exhausted all attempts to send the data to the Kafka receiver.

Fix: In Cribl Stream 4.2.

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 Worker Group's total count.

Workaround: For each Worker Group, click the link in its **Pipelines** column to see the Worker Group's actual list of Pipelines.

Fix: In Cribl Stream 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 Worker 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 Worker 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 Stream 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 Stream to match on the certificate's SAN (`subjectAltName`) extension.

Fix: In Cribl Stream 4.0.

2022-08-20 – v.3.2.0–4.0.3 – Deleting or modifying default Mapping Rule reclassifies Cribl-managed Cribl.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 Cribl.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 Stream 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 Stream 4.0.

2022-08-25 – Versions 3.4.1 through 4.4.4 – 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: In Cribl Stream 4.5.

2022-08-16 – All versions through 4.3.0 – Splunk HEC shows higher outbound data volume than other Splunk Destinations [CRIBL-11922]

Problem: Events sent to the Splunk HEC Destination will show higher outbound data volume than the same events sent to the Splunk Single Instance or Splunk Load Balanced Destinations, which use the S2S binary protocol.

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 Stream 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 Stream 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 Stream 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 Stream 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 Stream'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) Cribl.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 Stream 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 Stream 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 Stream 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 Stream 4.0.

2022-07-20 – v3.5.0–3.5.1 – Manage Worker Nodes page fails to lazy-load Workers when scrolled [CRIBL-11474]

Problem: Scrolling the **Manage Worker Nodes** page fails to load more than 50 Worker Nodes.

Fix: In Cribl Stream 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 rerunning Windows installers on a system that hosts a previous Cribl version as a managed Edge Node, the installer does not read the distributed mode or other settings. The new version might install as a Leader instance.

Workaround: Run the new version's installer using the same mode and other options you used when installing the preceding version.

Fix: In Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream is maximizing the queue.

Fix: Version TBD.

2022-06-29 – v.3.4.2–3.5.1 – Can't install a new Cribl Stream 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 Stream 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: Rerun the Windows installer on each Edge Node, specifying the same installation options/parameters you used when installing the preceding version.

Fix: In Cribl Stream 4.3.

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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 Stream version.

Fix: In Cribl Stream 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 Stream's host.

Workaround: Install `git` on the Leader or single instance. On affected Worker Nodes, install `git`, followed by a `git init` and an empty push to a new `master` branch. A simpler workaround for Worker Nodes is to just enable remote access (Stream, Edge) from the Leader.

Fix: In Cribl Stream 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 Stream 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 Stream 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 Stream 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 Stream 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 – 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.

2022-04-19 – 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 Stream 3.5.

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 Stream 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 Stream 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 Stream 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 Stream.

Fix: In Cribl Stream 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 Stream 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 Stream 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 Stream 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–Worker Group toggles are all set to `No` by default.

Workaround: On the **Manage Worker Groups** 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 Worker Group.

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 Worker Nodes [CRIBL-8938]

Problem: The Monitoring page displays an error where distributed environments have more than 30 Worker 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 Stream 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 Stream 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–Worker Group toggles on the **Manage Worker Groups** 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 Worker Group.

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.

2022-02-17 – 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.

2022-02-17 – 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.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.

2022-02-10 – 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 Stream 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.

2022-01-18 – 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.

2022-01-18 – 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.

2022-01-11 – 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.

2022-01-04 – 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 Stream 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.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.

2021-11-17 – 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.

2021-11-16 – 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 Stream 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 Stream/LogStream 3.2.1.

2021-10-22 – 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.

2021-10-11 – 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 Stream/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.

2021-09-30 – 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 Stream 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 through Current – 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**<img class="inline" src="/images/page/help-button.png" alt="Help link" width=" width="10%"" height="10%">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.

21.2. Working With Cribl Support

If you run into issues with Cribl Stream, please first check our Known Issues page for recommended resolutions or workarounds. For questions not addressed there, this page outlines how to engage with the Cribl Support staff to resolve problems as quickly as possible.

Contacting Cribl Support

You can contact Cribl Support in multiple ways, outlined below:

- Email
- Community Slack
- Cribl Curious
- Support Portal (login required)

The best method depends on your reason for contacting Support. For quick, question-and-answer discussions, Community Slack or Cribl Curious might be sufficient. For in-depth discussions, or to troubleshoot technical issues, create a support case for better tracking and exchange of information.

Creating a Support Case

You have two options for opening a support case: via email or via the Support Portal (access here, details below).

If you exchange diags or other files with us, note that all file-transfer methods **except** for email are encrypted.

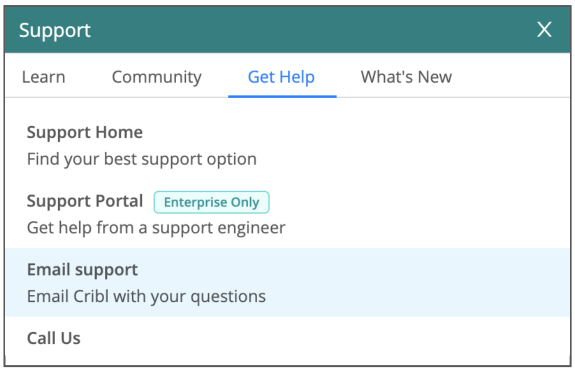
Email communication imposes a total message-size limit of 25 MB (after MIME-encoding attachments). The Support Portal allows file attachments of up to 100 MB each.

Contact via Email

The simplest way to engage with Support is to email support@cribl.io, with the information outlined below. This will automatically open a case for us to track, and will send you an auto-confirmation email that includes your case number. A Support engineer will then contact you to begin troubleshooting.

Email via Support Drawer

Within Cribl Stream's UI, you can click the top-right **Support** button and, in the resulting **Support** drawer, select the **Get Help** upper tab. Then click **Email support**. This will create a new email to us in your email client.



Email from the UI

Contact via Community Slack

Our Support staff monitors Cribl's Community Slack for any issues customers are experiencing: https://cribl-community.slack.com/. If you are not already registered for our Community Slack, please register at https://cribl.io/community/ to get started.

Slack might not get you the same timely response as an email, but it's a great way to get questions about Cribl Stream answered by a wide range of Cribl insiders, and expert peer users, who enjoy sharing their knowledge of the product. Check out individual channels dedicated to feature requests, docs, and other concerns.

Private Slack Channels

Our Enterprise customers have their own private channels where they can communicate directly with their Cribl account team.

Contact via Cribl Curious

Cribl Curious is a threaded community Q&A site. Our support staff monitors it, as do other Cribl teams. As with Community Slack, it's best for issues that aren't critical or urgent.

Contact via Support Portal

The Cribl Support Portal is available to our licensed customers. It facilitates encrypted transfer of large files, and maintains support cases' history so that you can easily review them.

The Support Portal uses single sign-on (SSO) through an integration with Cribl.Cloud accounts. So as part of Support Portal signup, you'll receive an invitation to create a Cribl.Cloud account, if you don't already have one. You have no obligation to use your Cribl.Cloud Organization for purposes other than SSO, if it doesn't meet your needs.

For details about navigating Cribl.Cloud, see the Cribl.Cloud.

The Support Portal enables two types of users: Standard and Admin. You can have up to four user logins per customer account, one of which can be (but is not required to be) an Admin user.

Standard Versus Admin Users

Standard users can:

- Create cases.
- Manage cases.
- Search cases.
- Upload files to cases.
- Access other Cribl resources via their Cribl.Cloud portal.

Admin users can do all of the above, plus:

- View all cases on the customer's account.
- Edit case information, post-creation.
- Invite Standard users to the portal.

Signing Up

You'll need to receive a Cribl.Cloud invitation from Cribl or your Support Portal Admin, and follow the directions in the email to sign up. Both scenarios are summarized below. But first, a word from our sponsor, us:

Video Tutorial: Signing Up, Submitting Cases

This video walks you through signing up for the Support Portal, and then submitting cases with enough detail for Cribl Support to rapidly help you.

Accessing the Cribl Support Portal

Standard User Signup

To access the Cribl Support Portal if you already have a Cribl.Cloud account:

1. Contact Cribl Support to request a user login to the Support Portal.
2. Cribl Support will send an invitation to you via email.
3. Follow the link in the email to sign up, using the same email address at which you received the invitation.
4. Log in with your existing Cribl.Cloud account.
5. Once logged in, you can create support cases, view any of your open or closed cases, etc.

To access the Cribl Support Portal without an existing Cribl.Cloud account:

1. Contact Cribl Support at support@cribl.io to request to be added. Cribl Support will email you an invitation. (Your account’s Support Portal Admin has the capability to invite you too.)
2. Follow the link in the email to sign up, using the same email address at which you received the invitation.
3. Register your new Cribl.Cloud account.
4. Then log into https://portal.support.cribl.io with your new Cribl.Cloud account.
5. Once logged in, you can create support cases, view any of your open or closed cases, etc.

Admin User Signup

To access the Cribl Support Portal as an Admin user:

1. Contact Cribl Support to request Admin access.
2. If your customer account already has an Admin user, the current Admin must also contact Cribl, requesting to transfer the account's sole Admin user role to you.
   The request from the current Portal Admin must originate from the email address we have on file for that Admin user. We can tell you who the current Admin is, if you do not know.
3. If you have an existing Cribl.Cloud account, then Cribl Support will promote your account to Admin. If you do not yet have a Cribl.Cloud account, Cribl Support will email you a Support Portal signup invitation. Follow the link in the email to accept the invitation, using the same email address at which you received the invitation.

Submitting Cases via the Portal

All support cases must be submitted by an individual account (not a shared or group account). However, there are broader notification options.

When you create a new case, note the two fields for **Related Teammates**. Each name entered here must be an existing Contact on your customer account. If you want to enter one of your own organization's **group** email addresses, contact Cribl Support to create a contact entry for that address.

Relevant Information We Need

When contacting Cribl Support via any means, please provide as many of the following details as you can – the more, the better:

- Your name.
- Preferred contact method (phone or email).
- Cribl Stream version number affected.
- Description of the issue you're having.
- What's the issue's scope? (Leader Node, specific Worker Nodes or Worker Groups, or entire deployment; number of nodes impacted).
- When did the issue begin, or when was it first noticed?
- Did you make any known changes around that time? (Upgrade, config change, network change, etc.).
- Diags for one or more affected systems (the Leader Node does not process data, so typically, only diags from Workers are necessary).
- Sample event data for testing Pipeline issues (provide a text file, rather than screenshots).

- Any troubleshooting steps that you've already taken.

Pulling Diags

Providing us diags from your environment will speed up the time to resolution. For instructions on how to pull a diag file, see Diagnosing Issues.

With Cribl Stream (LogStream) 2.4.1 or later, and Cribl Edge, the diag is uploaded from the browser, rather than from the Node. This means that your Worker Nodes do not need internet access.

With LogStream 2.4.0 or earlier, you'll need to transfer the diags from your Worker Nodes.

> Diag bundles for Leader Nodes do not include diag bundles for any Worker Nodes.
>
> If you would like to upload your diag file via the GUI or CLI, you'll need outbound internet access to the upload URL (https://diag-upload.cribl.io), plus a valid support case number (provided in your case confirmation email).

Diag Workarounds

If your organization does not permit outbound access to https://diag-upload.cribl.io to upload from within Cribl Stream, you can also submit diags directly via the Support Portal (login required).

If none of these options work with your organization's policies, please work directly with your Support engineer to find a solution.

21.3. Diagnosing Issues

To help diagnose Cribl Stream problems, you can share a diagnostic bundle with Cribl Support. The bundle contains a snapshot of configuration files and logs at the time the bundle was created, and gives troubleshooters insights into how Cribl Stream was configured and operating at that time.

What's in the Diagnostic Bundle

The following subdirectories (and their contents) of `$CRIBL_HOME` are included:

- `.../default/*`
- `.../local/*` – except for `/local/cribl/auth/`, to exclude sensitive files.
- `.../log/*`
- `.../groups/*`
- `.../state/jobs/*` – will return all jobs if left empty.

As a security measure, the bundle excludes all `.crt`, `.pem`, `.cer`, and `.key` files from all `$CRIBL_HOME` subdirectories.

Creating and Exporting a Diagnostic Bundle

If you're managing your own Cribl Stream deployment (single-instance Stream/Edge or distributed Stream/Edge), you can create and securely share bundles with Cribl Support, either from the UI or via the CLI.

> With a Cribl.Cloud deployment, create bundles and securely share them with Cribl Support directly from the UI. In the Using the UI instructions below, **Send to Cribl Support** will always be enabled, and the **Export**/download option will be unavailable.

Requirements

Using either medium, you'll need outbound internet access to `https://diag-upload.cribl.io` and a valid support case number. The above URL works only when using the `cribl diag` command or uploading using the Cribl Stream UI. (So connecting directly to it with your web browser will fail.)

Open your support case before sending a diagnostic bundle to Cribl Support. Then, when creating the bundle, be sure to use the case number provided. Case numbers are 8 digits in length, and you must include any leading zeros. For example, `00001234`.

Using the UI

Depending on the issue you're debugging, you can create a bundle either from the Leader or from an individual Worker Node's host.

From the Leader

To create a bundle from the Leader, go to **Settings** > **Global Settings** > **Diagnostics** > **Diagnostic Bundle** and click **Create Diagnostic Bundle**.

- To share the bundle with Cribl Support, toggle **Send to Cribl Support** to `Yes`. Enter your case number, and then click **Export**.
- To download the bundle locally to your machine (on-prem only), click **Export**.

From Worker Nodes

To create a bundle from individual Worker Nodes, you must be able to access the Worker Node's UI. You can easily use the Leader to get authenticated access to each Worker Node, if you first enable the parent Worker Group's UI Access (Stream/Edge)(Teleporting) setting:

1. Go to **Manage** > **Worker Groups**.
2. Make sure the relevant Worker Group's **UI Access** toggle is on.
3. Click the Worker Group's **Worker Node** link.
4. Click the link for the Worker Node you're debugging.
5. Look for the purple border to confirm that you've teleported to that Worker Node.
6. Click **Worker Node Settings** (top right) > **Diagnostics** (lower left) > **Diagnostic Bundle**, and click **Create Diagnostic Bundle**.
7. Use the appropriate **Export** options listed above for the Leader.

On customer-managed/on-prem deployments, new and previously created bundles are stored in `$CRIBL_HOME/diag`. They're also listed in the UI, where you can re-download them or share them with Cribl Support.

UI Options

The **Create & Export Diag Bundle** modal provides these additional options:

- **Rename .js files** toggle (defaults to `Yes`): Appends `.txt` to JavaScript files in the bundle.
- **Include metrics** toggle (defaults to `Yes`): Includes metrics in the bundle.

- **Include git** toggle (defaults to `Yes`): Includes the last 14 days of git log in the bundle, identifying each commit's message, author, timestamp, and files changed.
- **Latest number of jobs to include** field (defaults to empty = unlimited).

Cribl Support will advise you on changing the defaults here. Disabling **Include metrics** can reduce the diag bundle's size, where necessary.

Using the CLI

To create a bundle using the CLI, use the `diag` command. (This option is on-prem only.) Depending on the issue you're debugging, you can create a bundle either from the Leader's host or from an individual Worker Node's host.

diag command CLI

```
# $CRIBL_HOME/bin/cribl diag
Usage: [sub-command] [options] [args]

Commands:
create       - Creates diagnostic bundle for this instance, args:
  [-d]                  - Run create in debug mode
  [-j]                  - Do not append '.txt' to js files
  [-t <maxIncludeJobs>] - Latest number of jobs to include in bundle
  [-M]                  - Exclude metrics from bundle
  [-g]                  - Exclude git log from bundle
heapsnapshot - Generate heap snapshot of a Cribl process, args:
  [-p <pid>] - The pid of the process to dump the heap snapshot
list         - List existing diagnostic bundles
send         - Send diagnostics bundle to Cribl Support, args:
   -c <caseNumber> - Cribl Support Case Number
  [-p <path>]      - Diagnostic bundle path (if empty then new bundle will be creat

## Creating a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag create
Created Cribl Stream/Edge diagnostic bundle at /opt/cribl/diag/ <product>-<hostname

## Creating and sending a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -c <caseNumber>
Sent Cribl diagnostic bundle to Cribl Support

## Sending a previously created diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -p /opt/cribl/diag/ <product>-<hostname>-<datetin
Sent Cribl diagnostic bundle to Cribl Support
```

For complete command syntax, see diag.

Including CPU Profiles

If Cribl Support asks you to grab CPU profiles of Worker Processes, follow these steps:

1. Use `top` or `htop` on the Worker Node to identify Worker Node PIDs consuming a lot of CPU.
2. See Measuring CPU or API Load for instructions on accessing the UI's **Profile** options (for your deployment type), and generating and saving profiles.
3. Find the Worker Processes matching the PIDs you identified above.
4. Click **Profile** on each. Start with the default 10-second **Duration**.

5. Once the profile is displayed, save it to a `.json` file. (See details at the above link.)
6. Repeat steps 3–5 for other CPU-intensive Worker Processes.
7. Upload the profile `.json` files to Cribl Support.

On an already CPU-starved Worker Node, profiling might fail with an error message, or just hang. In this case, you might need a few retries to get a successful profile.

Including Memory Snapshots

Cribl.Cloud does not support taking memory snapshots. This feature is for on-prem deployments only.

If you encounter a memory leak, Cribl Support might request a memory snapshot of a `cribl` process to better understand where the leak is occurring. To capture the memory snapshot:

1. Note the PID of the process of interest.
2. In a terminal, `cd` to the directory where Cribl Stream is installed:

   ```
   cd $CRIBL_HOME/bin
   ```

3. Run the `diag heapsnapshot` command with the PID you noted in step 1:

   ```
   ./cribl diag heapsnapshot -p <pid>
   ```

Cribl will capture the memory snapshot in `$CRIBL_HOME/diag/`.

Example output:

```
Heap-1690906031148-2458958.heapsnapshot
```

21.4. Common Errors And Warnings

This page lists common error and warning messages that you might find in Cribl Stream's internal logs and/or UI. It includes recommendations for resolving the errors/warnings. Messages are grouped by component (Sources, Destinations, etc.). Note that:

- We've excerpted only the salient part of the error message from each event. We haven't listed full events.
- Examples don't preserve case sensitivity from the original events.
- NodeJS serves as the Cribl Stream backend and has a large collection of well documented errors of its own. Some of system-level are listed below with the appropriate action to take. The remainder are documented at https://nodejs.org/docs/latest-v14.x/api/errors.html#errors_node_js_error_codes, specifically in the `Common system errors` section of that page.
- Where events are written to log files, they might reside in different logs variously devoted to data processing, cluster communication, or the REST API. (For details, see Types of Logs.) We note the log type where necessary.

Web UI

Where: In Cribl Stream's UI.

Error: Duplicate GUID

When you deploy and first run Cribl Stream software, it generates a GUID (globally unique identifier) and stores it in a `.dat` file located in `CRIBL_HOME/bin/`. For example:

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

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

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

Error: "Failed to fetch"

Cause: This occurs only when accessing **Data** > **Collectors**. It can occur with ad blockers, and occurs with the Brave browser (because the newly loaded page's URL includes the string `collector`).

Recommendation: If you have an ad blocker, allowlist the `https://<hostname>:9000/jobs/collectors` (single-instance) URL or the `https://<masternode>:9000/m/<group_name>/jobs/collectors` (Leader) URL.

Error: "No workers registered"

Cause: Occurs during a live data capture, when there are no Workers available to fulfill the capture request. This error can be triggered if you attempt to capture events too quickly following a commit & deploy.

Recommendation: There are two things you can try:

- If you recently committed and deployed changes, wait 30 seconds and retry the live capture.
- Navigate to **Manage > Workers** and confirm there are Workers currently registered with the Leader. Note that the error appears on a per-Group basis, which means that even if you have Workers registered, they may not be in the Group you're capturing live data from.

Error: "The Config Helper service is not available because a configuration file doesn't exist or the settings are invalid. Please fix it and restart Cribl Stream."

Cause: Occurs only on Leader Nodes. Each Worker Group relies on a config helper process that runs on its behalf on a Leader Node, and that process is not currently running. This error is usually triggered by restarting Cribl Stream on the Leader Node, and then trying to access one of these UI features before the Cribl Stream server is fully up:

- **Groups** or **Configure** link (3.x).
- **Worker Groups** menu (2.x).
- **Manage** > **Groups** or **Manage** > **Workers** tabs (4.x).

Recommendation: Try again in a few seconds. If you've made changes to the backend or filesystem permissions, stop Cribl Stream, and make sure the `cribl` user owns all relevant assets, before restarting:
`sudo chown -R cribl.cribl /opt/cribl`

Warning: "KMS saved, but secret exists in destination location. No data migrated."

Cause: This will appear when you attempt to save KMS Settings with a **Secret Path** entry that references a secret that already exists in HashiCorp Vault. It can also occur with OpenID Connect remote authentication. Cribl Stream has aborted the remote write to avoid overwriting an external shared secret.

Recommendation: Edit the Leader Node's `kms.yml` file to use `provider: local`. Then restart Cribl Stream to correct the path conflict.

Logs: "The number of Worker Processes has been limited by the license."

Cause: Workers use their local license until they connect to the Leader and download their current license data, which is why the message appears in Logs.

Recommendation: Ignore the message. Once the Worker checks in with the Leader and receives the production license, the rest of the Workers will start.

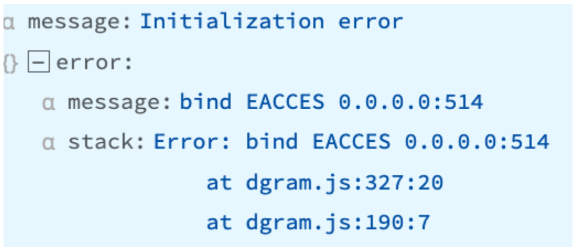
Sources (General)

Where: These events will be in the Worker process logs.

Error: "bind EACCES 0.0.0.0:514"

Cause: Cribl Stream doesn't have proper privileges to bind to the specified IP and port. Usually, this simply indicates that Cribl Stream is running as a non-root user, but a Source was accidentally configured to use a privileged port below 1024.

Recommendation: Check your Cribl Stream Sources' configuration. The error's `channel` field will indicate which input is affected. If you choose to enable access to the port range below 1024, see our configuration instructions for systemd and initd.



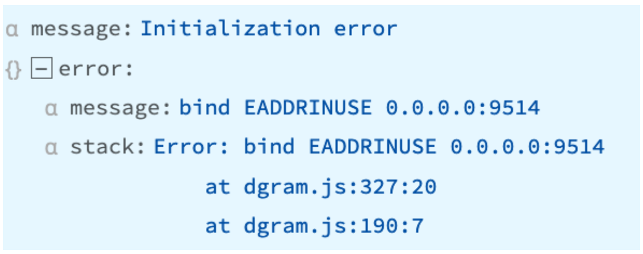
```
α message: Initialization error
{} ⊟ error:
   α message: bind EACCES 0.0.0.0:514
   α stack: Error: bind EACCES 0.0.0.0:514
                at dgram.js:327:20
                at dgram.js:190:7
```

EACCES error

Error: "bind EADDRINUSE 0.0.0.0:9514"

Cause: The interface and port combination is already in use by another process, which might or might not be Cribl Stream.

Recommendation: Check your Cribl Stream Sources' configuration. The error's `channel` field will indicate which Source is affected, but this error means that one or more other inputs are also using the same IP/port combination.



```
ɑ message: Initialization error
{} ⊟ error:
    ɑ message: bind EADDRINUSE 0.0.0.0:9514
    ɑ stack: Error: bind EADDRINUSE 0.0.0.0:9514
                at dgram.js:327:20
                at dgram.js:190:7
```

ADDRINUSE error

HTTP-based Source

Where: These events will be in the Worker Process logs.

Message: “Dropping request because token invalid”,“authToken”: “Bas...Njc=”" (v2.4.5 and later)

Cause: The specified token is invalid. Note the above message is logged only at debug level.

Kafka-based Source

Where: These events will be in the Worker Process logs.

Error: “KafkaJSProtocolError: Not authorized to access topics: [Topic authorization failed]”

Cause: The username does not have read permissions for the specified topic.

Splunk TCP Source

Where: These events will be in the Worker Process logs.

Error: “connection rejected” with a `reason` of “Too many connections”

Cause: The maximum number of active Splunk TCP connections has been exceeded per worker process. The default is 1000.

Recommendation: In the Splunk TCP input's Advanced Settings configuration, increase the Max Active Connections value, set it to 0 for unlimited, and/or increase the # of worker processes the Worker Node(s) are using..

Splunk HEC Source

Where: These events will be in the Worker Process logs.

Error: "Malformed HEC event"

Cause: The event is missing both `event` and `fields` fields.

Error: "Invalid token"

Cause: Auth token(s) are defined, but the token specified in the HEC request doesn't match any defined tokens, therefore it's invalid.

Error: "Invalid index"

Cause: The Splunk HEC Source's **Allowed Indexes** field is configured with specific indexes, but the client's HTTP request didn't specify any of them.

Error: "{"text":"Server is busy","code":9,"invalid-event-number":0}"

Note: This is not logged in Cribl Stream, but would be found in the response payload sent to your HEC client.

Cause: "Server is busy" is the equivalent of a 503 HTTP response code. The most likely cause is the **Max active requests** setting in the HEC input's Advanced Settings is insufficient to service the number of simultaneous HEC requests. Increase the value and monitor your clients to see if the 503 response is eliminated.

TLS Errors

Where: These events can be in Worker Process or API logs on the Workers or Leader, depending on whether the issue is associated with a Source or Destination, etc.

Error: "certificate has expired"

Cause: **Validate server certs** or **Validate client certs** is enabled, and the peer's certificate has expired.

Recommendation: Disable **Validate server certs** or **Validate client certs** (depending on whether Cribl Stream is serving as the client or server), so that encryption can still occur without authentication. Or renew the expired certificate.

Error: “Client network socket disconnected before secure TLS connection was established”

Cause: Typically caused by a TLS protocol version mismatch between the client and server.

Recommendation: Verify that client’s and server’s TLS settings use the same minimum/maximum TLS version.

Error: “Unable to get local issuer certificate”

Cause: Client can’t validate the server certificate’s CA (i.e., the issuer) because it doesn’t trust the CA cert. Or vice versa (server can’t validate client’s certificate CA) if mutual auth is enabled.

Recommendation: The CA certificates used by the server’s leaf certificate must be trusted by the client. See Configuring CA certs.

Error: “self signed certificate”

Cause: The client or server was presented with a self-signed cert from the peer, but that cert is not trusted.

Recommendation: The self-signed certificate must be trusted by the peer to which it’s presented. See Configuring CA certs.

Error: “Hostname/IP does not match certificate’s altnames"

Variations: “Hostname/IP does not match certificate’s altnames: host: <server hostname> is not in the cert’s list” or: “Hostname/IP does not match certificate’s altnames: IP: <server IP> is not in the cert’s list”

Cause: The server’s hostname/FQDN used on the client is not found in the CN or SAN attribute of the server’s certificate.

Recommendation: Examine the CN and/or SAN attribute, to see which names are listed that can be used as the server hostname/FQDN on the client. **CN values with spaces are not supported as hostnames/FQDNS.** If there isn’t a SAN attribute, then a new cert will need to be issued.

Error: “140244944713600:error:141E70BF:SSL routines:tls_construct_client_hello:no protocols available:”

Cause: The highest TLS protocol available by the client is still too low for the server to support.

Recommendation: Review the minimum/maximum TLS version settings on the client and server, to ensure that they overlap.

Error: “140251374995328:error:1408F10B:SSL routines:ssl3_get_record:wrong version number”

Cause: The client is using TLS but the server is probably not configured for TLS.

Recommendation: Review the TLS settings on the server.

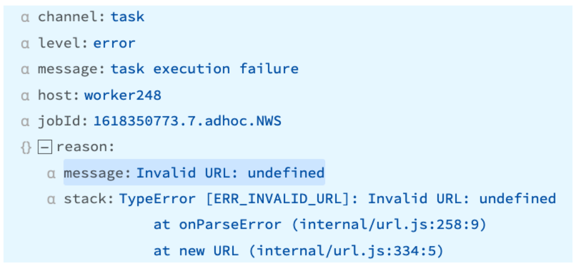
REST API Collector

Where: These events will be in the Worker Process logs.

Error: “reason.stack == `TypeError [ERR_INVALID_URL]: Invalid URL: https%3A%2F%2Ftype.fit%2Fapi%2Fquotes” at onParseError (internal/url.js:258:9)"

Cause: This is due to unnecessarily encoding the Discover/Collect URL.

Recommendation: Remove the encoding function for the URL. URLs will rarely need text be encoded and when they do it’s only the parts that need it that should be encoded, otherwise if the entire URL is encoded unnecessarily then errors like this will occur.



```
channel: task
level: error
message: task execution failure
host: worker248
jobId: 1618350773.7.adhoc.NWS
reason:
  message: Invalid URL: undefined
  stack: TypeError [ERR_INVALID_URL]: Invalid URL: undefined
             at onParseError (internal/url.js:258:9)
             at new URL (internal/url.js:334:5)
```

Invalid URL

Error: “statusCode: 429...Too many requests”

Cause: This response is triggered by rapidly repeated authentication requests from the Collector’s Discover and Collect phases. It’s especially likely when different Workers run multiple Collect tasks.

Recommendation: Gradually increase the **Login rate limit** threshold until you no longer receive this error response. On the Leader or a single-instance deployment, access this option at **Global Settings** > **General Settings** > **API Server Settings** > **Advanced**. In a distributed deployment, select **Manage**, select the Worker Group, and then navigate to **Group Settings** > **General Settings** > **API Server Settings** > **Advanced**.

AWS Sources/Destinations & S3-Compatible Stores

Where: These events will be in the Worker Process logs.

Error: "message":"Inaccessible host: 'sqs.us-east-1.amazonaws.com'. This service may not be available..."

Full Error Text: `"message":"Inaccessible host: 'sqs.us-east-1.amazonaws.com'. This service may not be available in the 'us-east-1' region.","stack":"UnknownEndpoint: Inaccessible host: 'sqs.us-east-1.amazonaws.com'. This service may not be available in the 'us-east-1' region.`

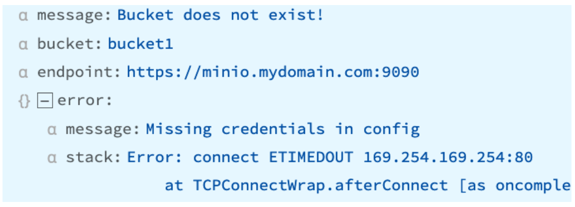
Cause: If this is persistent rather than intermittent then it could be caused by TLS negotiation failures. For example, AWS SQS currently does not support TLS 1.3. If intermittent then a network-related issue could be occurring such as DNS-related problems.

Error: "Missing credentials in config" or "stack:Error: connect ETIMEDOUT 169.254.169.254:80"

Cause 1: Can occur when **Authentication** is set to **Auto**, but no IAM role is attached.

Cause 2: Can occur on LogStream 2.4.4 or earlier, when an IAM role is attached to the EC2 instance, but the instance is using instance metadata v2.

Recommendations: Change to Manual Authentication; attach an IAM role; or if using IMDv2, switch to IMDv1 (if possible) or upgrade to LogStream 2.4.5 or later.



```
α message: Bucket does not exist!
α bucket: bucket1
α endpoint: https://minio.mydomain.com:9090
{} ⊟ error:
   α message: Missing credentials in config
   α stack: Error: connect ETIMEDOUT 169.254.169.254:80
                at TCPConnectWrap.afterConnect [as oncomple
```

Missing credentials in config

Error: “message”:“Bucket does not exist - self signed certificate

Example Error Text: `message: Bucket does not exist - self signed certificate stack: Error: self signed certificate at TLSSocket.onConnectSecure (node:_tls_wrap_1535:34) at TLSSocket.emit (node:events:513:28) at TLSSocket.emit (node:events:489:12) at TLSSocket._fini... Show more tmpPath: /opt/cribl/s3_bucket/cribl/2023/03/28/CriblOut-XTNGh5.0.json.tmp`

Cause 1: Can occur when the Destination uses a self-signed cert that has not yet been trusted by Cribl Stream.

Cause 2: Can occur when the user does not have appropriate permissions to view the bucket.

Cause 3: Can occur when the authentication token generated by your computer has a timestamp that is out of sync with the server’s time, resulting in an authentication failure.

Recommendations: Disable **Advanced Settings** > **Reject unauthorized certificates** or; get the sender’s cert and add it to the `NODE_EXTRA_CA_CERTS` path for validation; or verify that the user has appropriate permissions.

Destinations (General)

Where: These events will be in the Worker Process logs, unless otherwise noted.

Warning: “sending is blocked”

Cause: The Destination is blocking. This can occur with any TCP-based Destination, and is logged only after 1 second of blocking. This can also occur between Worker and Leader when the Worker can’t connect to the Leader Node to send metrics data. When triggered by cluster communication, the warning will be in the Worker’s API log.

Warning: “exerting backpressure” (v2.4.0-2.4.1)

Cause: The Destination is blocking. This message is logged immediately upon detecting backpressure, for Destinations using any protocol. Some backpressure is normal when measured over timescales under 1 second, therefore this message can appear quite frequently, and is not indicative of a problem (which is why it’s a warning).

Warning: “begin backpressure” and “end backpressure” (v2.4.2 and later)

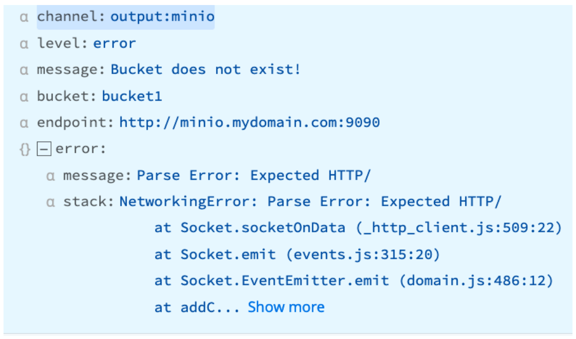
Cause: The Destination is blocking. Like the “sending is blocked” message, “begin backpressure” is logged only after 1 second of blocking. Unlike the “exerting backpressure” message, it is logged only once while backpressure is occurring (at the start), and it will always be followed by the “end backpressure” message.

MinIO Destination

Where: These events will be in the Worker Process logs.

Error: “Parse Error: Expected HTTP/”

Cause: The Worker is trying to use HTTP, but the server is expecting HTTPS.


```
a channel: output:minio
a level: error
a message: Bucket does not exist!
a bucket: bucket1
a endpoint: http://minio.mydomain.com:9090
{} ⊟ error:
   a message: Parse Error: Expected HTTP/
   a stack: NetworkingError: Parse Error: Expected HTTP/
                at Socket.socketOnData (_http_client.js:509:22)
                at Socket.emit (events.js:315:20)
                at Socket.EventEmitter.emit (domain.js:486:12)
                at addC... Show more
```

Parse Error: Expected HTTP

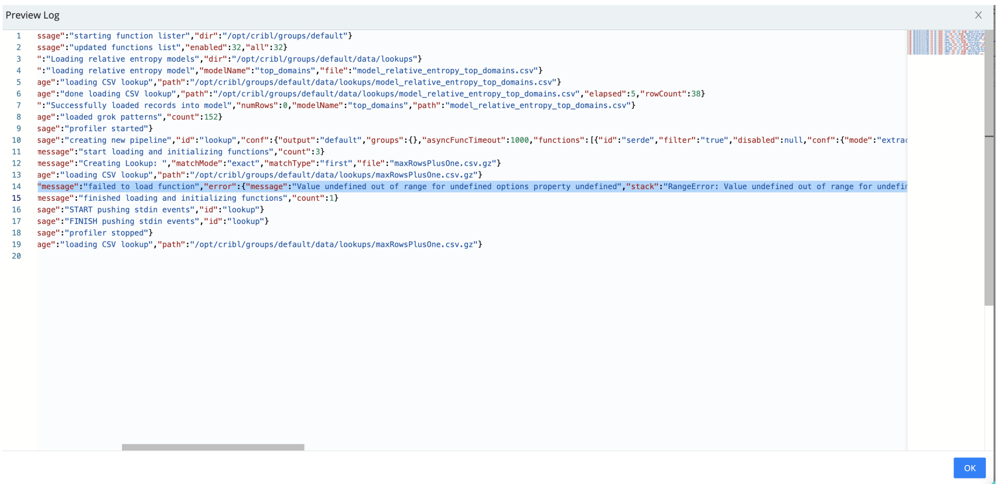
Pipelines/Functions

Where: These events will be in the Worker Process logs.

Error: “failed to load function…Value undefined out of range…”

Cause: A Lookup Function attempted to load a lookup table that exceeded Node.js’ hard size limit of 16,777,216 (i.e., 2^{24}) rows.

Recommendation: Split the lookup table to smaller tables, or use the Redis Function.



```
ssage":"starting function lister","dir":"/opt/cribl/groups/default"}
ssage":"updated functions list","enabled":32,"all":32}
":"Loading relative entropy models","dir":"/opt/cribl/groups/default/data/lookups"}
":"loading relative entropy model","modelName":"top_domains","file":"model_relative_entropy_top_domains.csv"}
age":"loading CSV lookup","path":"/opt/cribl/groups/default/data/lookups/model_relative_entropy_top_domains.csv"}
age":"done loading CSV lookup","path":"/opt/cribl/groups/default/data/lookups/model_relative_entropy_top_domains.csv","elapsed":5,"rowCount":38}
":"Successfully loaded records into model","numRows":0,"modelName":"top_domains","path":"model_relative_entropy_top_domains.csv"}
age":"loaded grok patterns","count":152}
sage":"profiler started"}
sage":"creating new pipeline","id":"lookup","conf":{"output":"default","groups":{},"asyncFuncTimeout":1000,"functions":[{"id":"serde","filter":"true","disabled":null,"conf":{"mode":"extrac
message":"start loading and initializing functions","count":3}
message":"Creating Lookup: ","matchMode":"exact","matchType":"first","file":"maxRowsPlusOne.csv.gz"}
age":"loading CSV lookup","path":"/opt/cribl/groups/default/data/lookups/maxRowsPlusOne.csv.gz"}
"message":"failed to load function","error":{"message":"Value undefined out of range for undefined options property undefined","stack":"RangeError: Value undefined out of range for undefir
message":"finished loading and initializing functions","count":1}
sage":"START pushing stdin events","id":"lookup"}
sage":"FINISH pushing stdin events","id":"lookup"}
sage":"profiler stopped"}
age":"loading CSV lookup","path":"/opt/cribl/groups/default/data/lookups/maxRowsPlusOne.csv.gz"}
```

Oversized lookup table error

Diag Command

Where: On stdout

Warning: “You are running Cribl Stream CLI as user=root, while the binary is owned by the user=cribl.”

Full Text: “WARNING: You are running Cribl Stream CLI as user=root, while the binary is owned by the user=cribl. This may change the ownership of some files under CRIBL_HOME=/opt/cribl. Please make sure all files under CRIBL_HOME=/opt/cribl are owned by the user=cribl.”

Cause: This is caused by improper ownership on `$CRIBL_HOME`, and will cause some files to be missing from the diag.

Recommendation: Execute the `chown` command on the entire `$CRIBL_HOME` directory, so that everything can be owned by the proper user. Afterward, run the `./cribl diag create` command again.

Cluster

Where: These events will be in the Workers’ API logs.

Error: “access denied”

Cause: The Worker’s authtoken (located in `$CRIBL_HOME/local/_system/instance.yml`) is missing or doesn’t match the Leader’s.

21.5. Git Push Errors

This page anticipates common errors you might see in Cribl Stream's UI, or in the `git` CLI, when pushing a commit.

Failed to Push Some Refs

Your first push to a remote repo might fail with one of several `failed to push some refs` errors.

As a first step in debugging these errors, edit the `$CRIBL_HOME/.git/config` file to make sure that its `name` and `email` key values match the credentials you've set on your repo provider or git server.

Also make sure that the `remote "origin"` key value matches the remote you set when you connected to the remote repo. This example shows all three keys, with placeholder values:

```
[user]
        name = <your-login-name>
        email = <email@example.com>
[remote "origin"]
        url = https://<user-name>:<token>@github.com/<username>/<repo-name>
```

Next, verify the remote repo from the command line, as follows:

```
cd $CRIBL_HOME/.git
git remote -v
```

In response, `git` should echo your configured remote twice – once for `fetch` and once for `push` operations.

If all of the above settings are correct, the `push` is very likely blocking because the remote repo has some commit history, or was simply created with a `readme.md` file. For command-line instructions to remedy this – by syncing your local repo to its remote – see GitHub's Dealing with Non-Fast-Forward Errors topic.

Large Files Detected

A push command might also trigger "large file" warnings or, more seriously, errors of this form (CLI/GitHub example):

```
remote: warning: File data/lookups/geo.mmdb is 60.12 MB; this is larger than GitHul
remote: error: GH001: Large files detected. You may want to try Git Large File Stor
remote: error: Trace: [##############################################################
remote: error: See http://git.io/iEPt8g for more information.
remote: error: File groups/default/data/lookups/largelookup.csv is 313.91 MB; this
```

Cribl recommends adding such large files to `.gitignore`, to exclude them from subsequent `push` commands. As the above examples show, typical culprits are large `.csv` or `.mmdb` lookup files. A simple option is to place these files in a `$CRIBL_HOME` subdirectory that's already listed in `.gitignore` – for details, see Managing Large Lookups.

Other available workarounds include staging such files **outside** `$CRIBL_HOME`, or using plugins to accommodate the large files. For GitHub-specific options, see Working with Large Files.

See Also

- Git Remote Repos with Trusted CAs

21.6. Git Remote Repos & Trusted CAs

If you are using an internal Git server, a self-signed certificate might prevent Cribl Stream from successfully pushing commits to the origin. You might see errors like these when pushing (or pulling) via the CLI:

```
SSL certificate problem: self signed certificate in certificate chain
SSL certificate problem: unable to get local issuer certificate
```

Resolving the Errors

To ensure that Git trusts your self-signed certificate, follow these steps:

1. Obtain the certificate chain (root, intermediates, and leaf) for the Git server.
2. As the `cribl` user, run this command: `git config http.sslCAInfo /path/to/certs.pem`
3. Test with this command: `git push origin` Verify that this throws no errors.

Obtain the Certificate Chain (TLS/SSL)

Use these steps to enable Worker-to-Leader mutual authentication:

A. Validate the Client Certs

If you are using an internal certificate authority, obtain a copy of the CA public certificate, then add it to `/etc/systemd/system/cribl.service`:

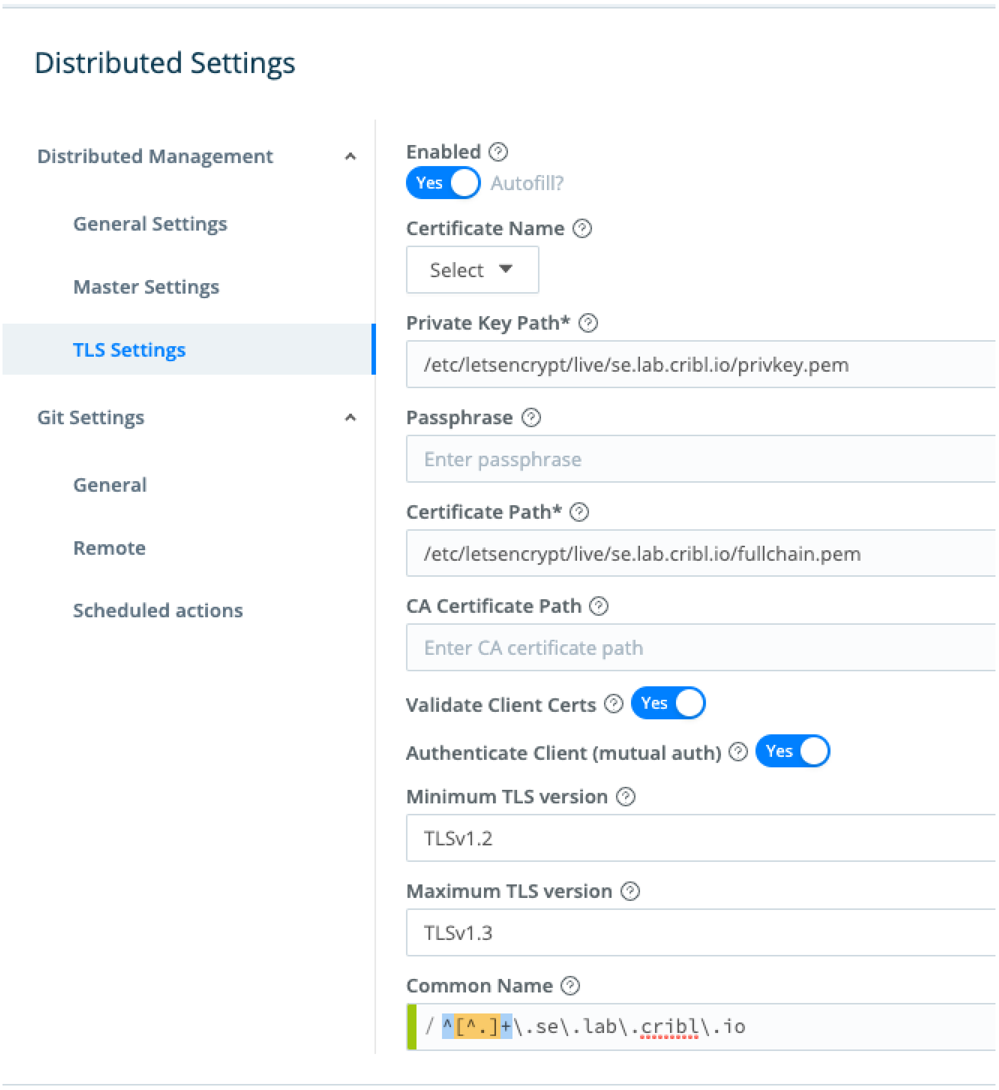
```
...
[Service]
Environment="NODE_EXTRA_CA_CERTS=/opt/cribl/local/cribl/auth/certs/ca.pem"
...
```

For details, see CA Certificates and Environment Variables.

B. Simplify the Common-Name Regex

The common-name regex (if required) should omit the `CN=` at the beginning of the **Common Name** field. The example below will match all immediate subdomains of `se.lab.cribl.io`, like `madsci.se.lab.cribl.io`.

If you disable **Validate Client Certs**, Cribl Stream will match only on common names.


Distributed Settings

Enabled: Yes
Certificate Name: Select
Private Key Path*: /etc/letsencrypt/live/se.lab.cribl.io/privkey.pem
Passphrase: Enter passphrase
Certificate Path*: /etc/letsencrypt/live/se.lab.cribl.io/fullchain.pem
CA Certificate Path: Enter CA certificate path
Validate Client Certs: Yes
Authenticate Client (mutual auth): Yes
Minimum TLS version: TLSv1.2
Maximum TLS version: TLSv1.3
Common Name: / ^[^.]+\.se\.lab\.cribl\.io

Common Name example

C. Extract SSL Certificate Info

As in this example:

```
openssl x509 -in certificate.pem -text -noout
```

D. Dump the Certificate Chain from the Server

As in this example:

```
echo "" | openssl s_client -host www.google.com -port 443 -showcerts 2>&1 | sed -n
```

21.7. Sample Logs For Login Scenarios

This page lists sample event series that Cribl Stream records in common login scenarios. Cribl Stream captures event series for both successful and failed login attempts. Note that these samples omit the following repeating fields that are captured in each log:

- `cid:api`
- `channel:cribl`
- `level:info`

Successful Authentication Using the `memberOf` Attribute

```
{"time":"2022-03-30T03:09:55.569Z","message":"Running LDAP search","searchBase":"dc=mydomain,dc=com","filter":"(sAMAccountName=admin)","options":{"scope":"sub","filter":"(sAMAccountName=admin)"}}
```

```
{"time":"2022-03-30T03:09:55.575Z","message":"LDAP user search results","count":1,"username":"REDACTED","user":{"dn":"CN=admin,CN=Users,DC=mydomain,DC=com","memberOf":["CN=admingrp,CN=Users,DC=mydomain,DC=com"]}}
```

```
{"time":"2022-03-30T03:09:55.582Z","channel":"LDAPMapper","level":"debug","message":"External groups found","groups":["admingrp","Users"]}
```

```
{"time":"2022-03-30T03:09:55.583Z","message":"Successful login","user":"admin","provider":"ldap:win2008ad1.mydomain.com:389"}
```

Successful Authentication With Fallback When a Bad Login Is Triggered

```
{"time":"2022-03-30T03:10:48.389Z","message":"Running LDAP search","searchBase":"dc=mydomain,dc=com","filter":"(sAMAccountName=admin)","options":{"scope":"sub","filter":"(sAMAccountName=admin)"}}
```

```
{"time":"2022-03-30T03:10:48.393Z","message":"LDAP user search results","count":1,"username":"REDACTED","user":
```

```
{"dn":"CN=admin,CN=Users,DC=mydomain,DC=com","memberOf":
["CN=admingrp,CN=Users,DC=mydomain,DC=com"]}}
```

```
{"time":"2022-03-30T03:10:48.399Z","message":"Attempting fallback to local
authentication","reason":"failed
login","provider":"ldap:win2008ad1.mydomain.com:389","user":"admin"}
```

```
{"time":"2022-03-30T03:10:48.400Z","message":"Successful
login","user":"admin","provider":"local"}
```

Failed Authentication with Fallback When a Bad Login Is Disabled

```
{"time":"2022-03-30T03:11:54.175Z","message":"Running LDAP
search","searchBase":"dc=mydomain,dc=com","filter":"
(sAMAccountName=admin)","options":{"scope":"sub","filter":"(sAMAccountName=admin)"}}
```

```
{"time":"2022-03-30T03:11:54.178Z","message":"LDAP user search
results","count":1,"username":"REDACTED","user":
{"dn":"CN=admin,CN=Users,DC=mydomain,DC=com","memberOf":
["CN=admingrp,CN=Users,DC=mydomain,DC=com"]}}
```

```
{"time":"2022-03-30T03:11:54.184Z","level":"warn","message":"Failed
login","user":"admin","provider":"ldap:win2008ad1.mydomain.com:389","details":
{"message":"80090308: LdapErr: DSID-0C0903A9, comment: AcceptSecurityContext error,
data 52e, v1db1\u0000 Code: 0x31","stack":"Error: 80090308: LdapErr: DSID-0C0903A9,
comment: AcceptSecurityContext error, data 52e, v1db1\u0000 Code: 0x31
```

Incorrect Proxy Bind Password With Fallback When a Fatal Error Is Triggered

```
{"time":"2022-03-30T03:04:16.400Z","level":"warn","message":"Authentication Provider
Error","provider":"ldap:win2008ad1.mydomain.com:389","fatal":{"message":"80090308:
LdapErr: DSID-0C0903A9, comment: AcceptSecurityContext error, data 52e, v1db1\u0000
Code: 0x31","stack":"Error: 80090308: LdapErr: DSID-0C0903A9, comment:
AcceptSecurityContext error, data 52e, v1db1\u0000 Code: 0x31\n at Function.parse
[snip]
```

```
{"time":"2022-03-30T03:04:16.401Z","message":"Attempting fallback to local
authentication","reason":"provider
```

```
error","provider":"ldap:win2008ad1.mydomain.com:389","user":"admin","error":
{"message":"80090308: LdapErr: DSID-0C0903A9, comment: AcceptSecurityContext error,
data 52e, v1db1\u0000 Code: 0x31","stack":"Error: 80090308: LdapErr: DSID-0C0903A9,
comment: AcceptSecurityContext error, data 52e, v1db1\u0000 Code: 0x31
```

Successful Authentication Without `memberOf` to Trigger Group Search but Without Group Memberships

```
{"time":"2022-03-30T03:15:21.410Z","message":"Running LDAP
search","searchBase":"dc=mydomain,dc=com","filter":"
(sAMAccountName=admin)","options":{"scope":"sub","filter":"(sAMAccountName=admin)"}}

{"time":"2022-03-30T03:15:21.416Z","message":"LDAP user search
results","count":1,"username":"REDACTED","user":
{"dn":"CN=admin,CN=Users,DC=mydomain,DC=com"}}

{"time":"2022-03-30T03:15:21.422Z","message":"Running LDAP
search","searchBase":"dc=mydomain,dc=com","filter":"
(member=CN=admin,CN=Users,DC=mydomain,DC=com)","options":{"scope":"sub","filter":"
(member=CN=admin,CN=Users,DC=mydomain,DC=com)"}}

{"time":"2022-03-30T03:15:21.424Z","message":"LDAP groups search
results","count":0,"opts":{"groupSearchBase":"dc=mydomain,dc=com","memberSearch":
{"memberField":"member","memberDN":"CN=admin,CN=Users,DC=mydomain,DC=com"}},"groups":
[]}

{"time":"2022-03-
30T03:15:21.425Z","channel":"LDAPMapper","level":"debug","message":"External groups
found","groups":[]}

{"time":"2022-03-30T03:15:21.427Z","message":"Successful
login","user":"admin","provider":"ldap:win2008ad1.mydomain.com:389"}
```

22. Third-Party Credits

Various components in Cribl Stream are built and enhanced with software under free or open source licenses. We thank those projects' contributors!

Package name	Version	License	Copyright