Cribl Stream Documentation Manual

Version: v3.5

Generated: 2022-10-13 13:45:40

1. Introduction

1.1. About Cribl Stream

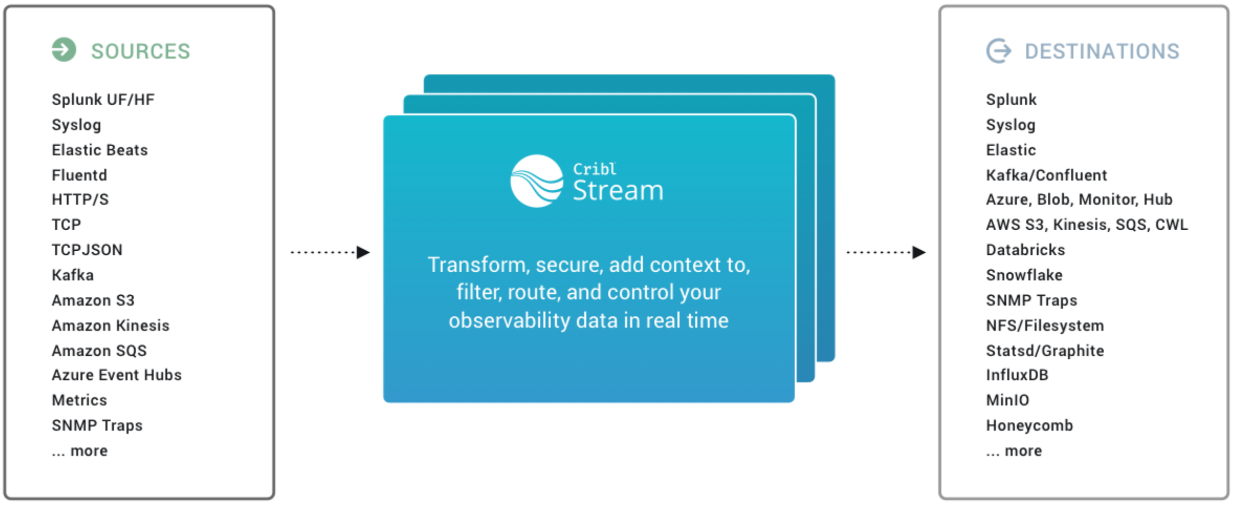
All the docs to 🐐 goat you started with Cribl Stream

- Download all docs as a PDF – v.3.5.3 | 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?

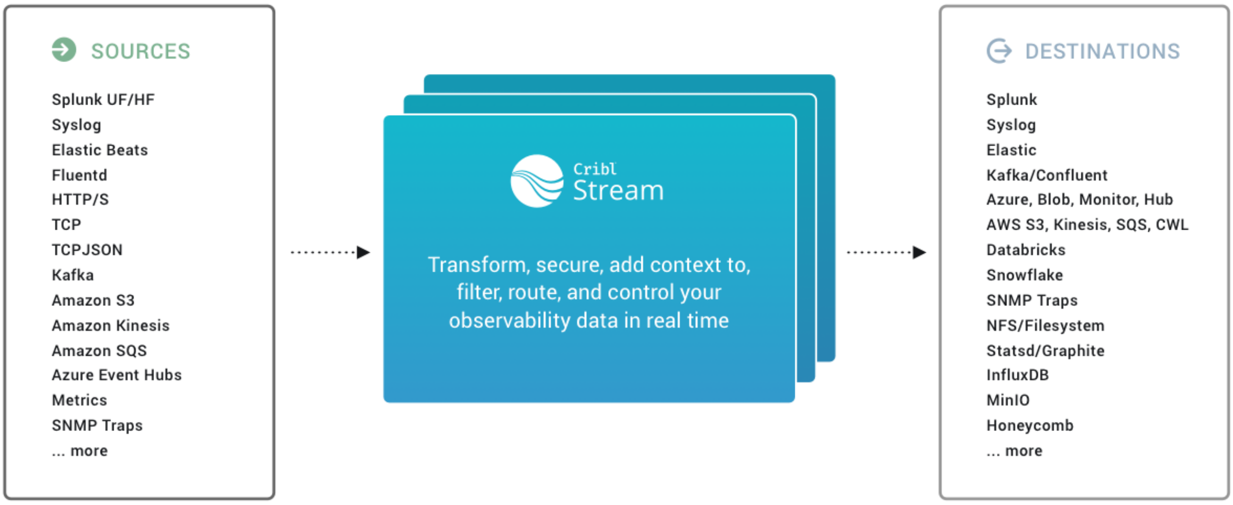
Cribl Stream is built for administrators, managers, and users of operational/DevOps and security intelligence products and services.

;

1.2. Basic Concepts

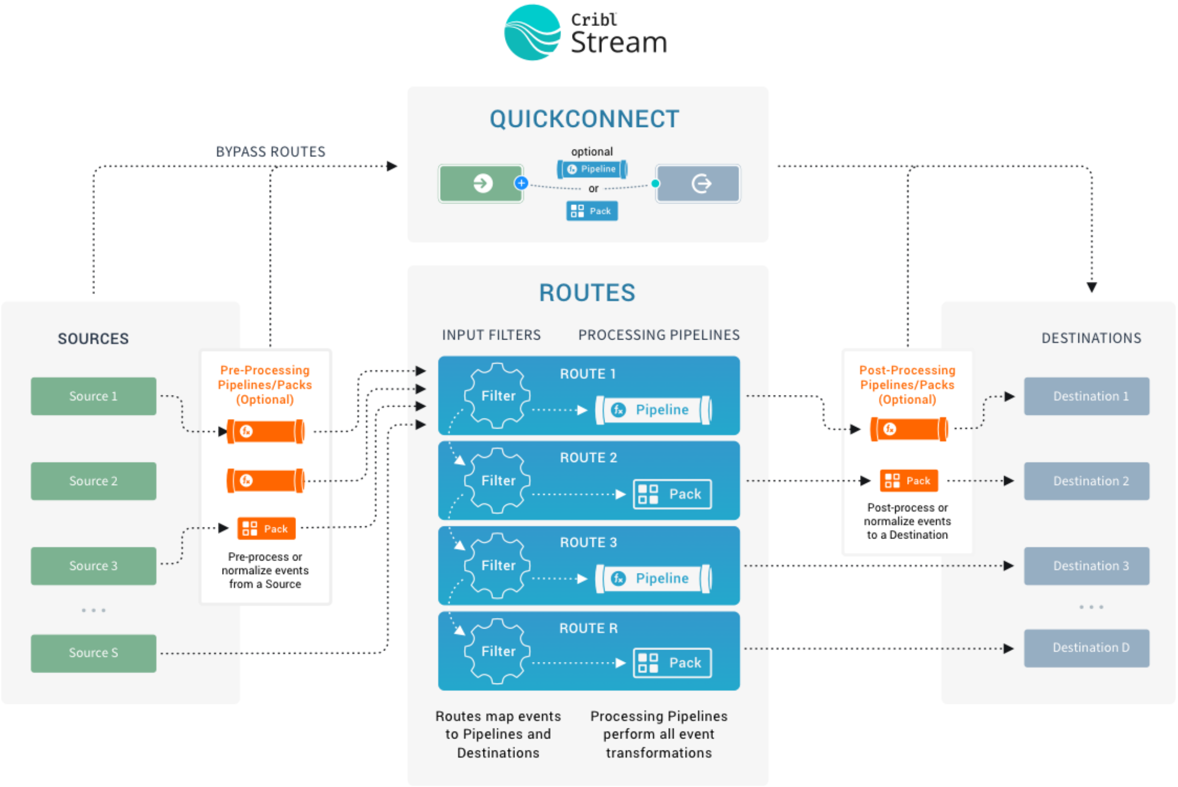
Notable features and concepts to get a fundamental understanding of Cribl Stream

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

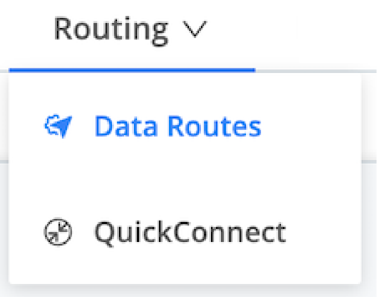
Sources are configurations that enable Cribl Stream to receive data from remote senders (Splunk, TCP, Syslog, etc.), or to collect data from remote file stores or the local machine.

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

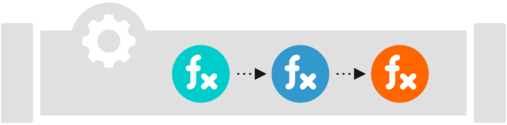
Routes evaluate incoming events against **filter** expressions to find the appropriate Pipeline to send them to. Routes are **evaluated in order**. Each Route can be associated **with only one** Pipeline and one output (configured as a Cribl Stream **Destination**).

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.

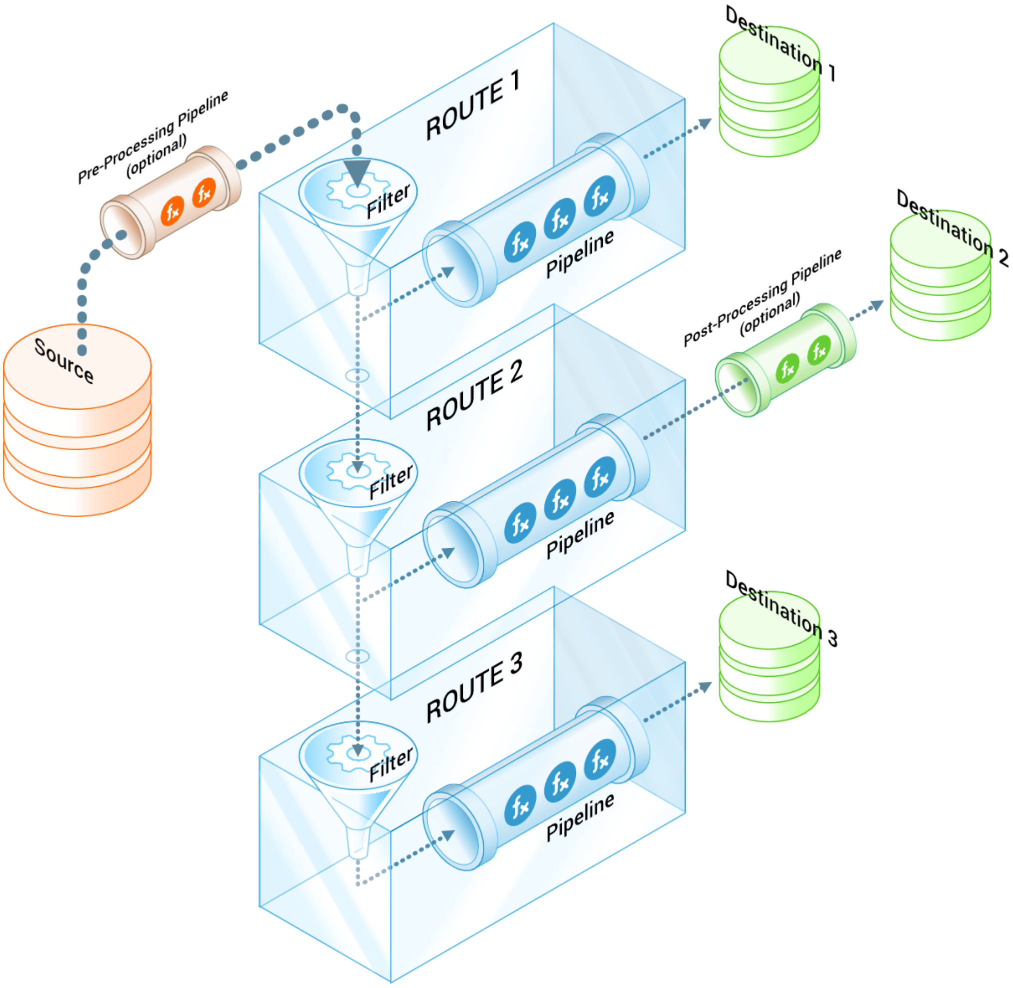
However, if you disable the Route's `Final` flag, one or more event **clones** will be sent down the associated Pipeline, while the original event continues down Cribl Stream's Routing table to be evaluated against other configured Routes. This is very useful in cases where the same set of events needs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

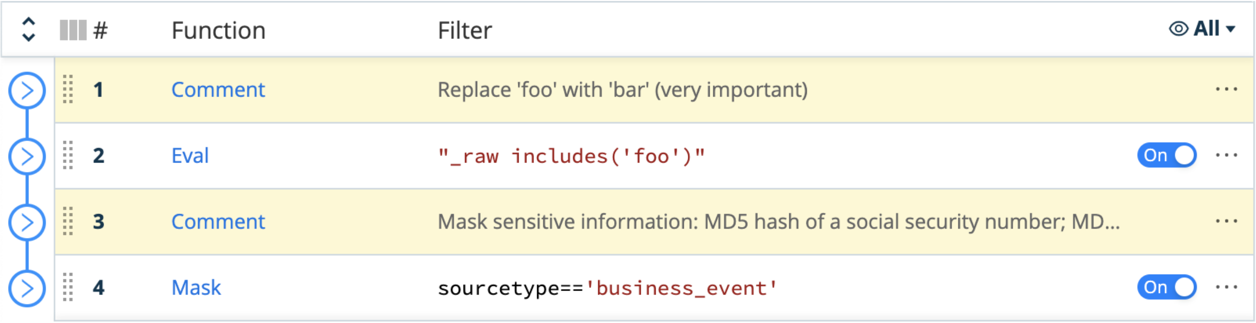


Pipelines categorized by position

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`.



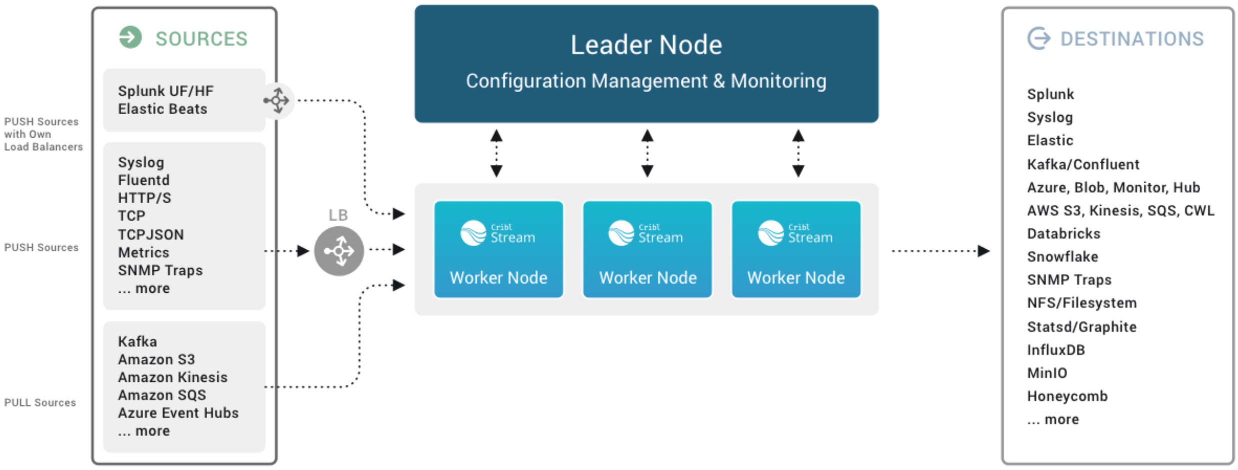
#	Function	Filter	
1	Comment	Replace 'foo' with 'bar' (very important)	
2	Eval	"_raw includes('foo')"	On
3	Comment	Mask sensitive information: MD5 hash of a social security number; MD...	
4	Mask	sourcetype=='business_event'	On

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

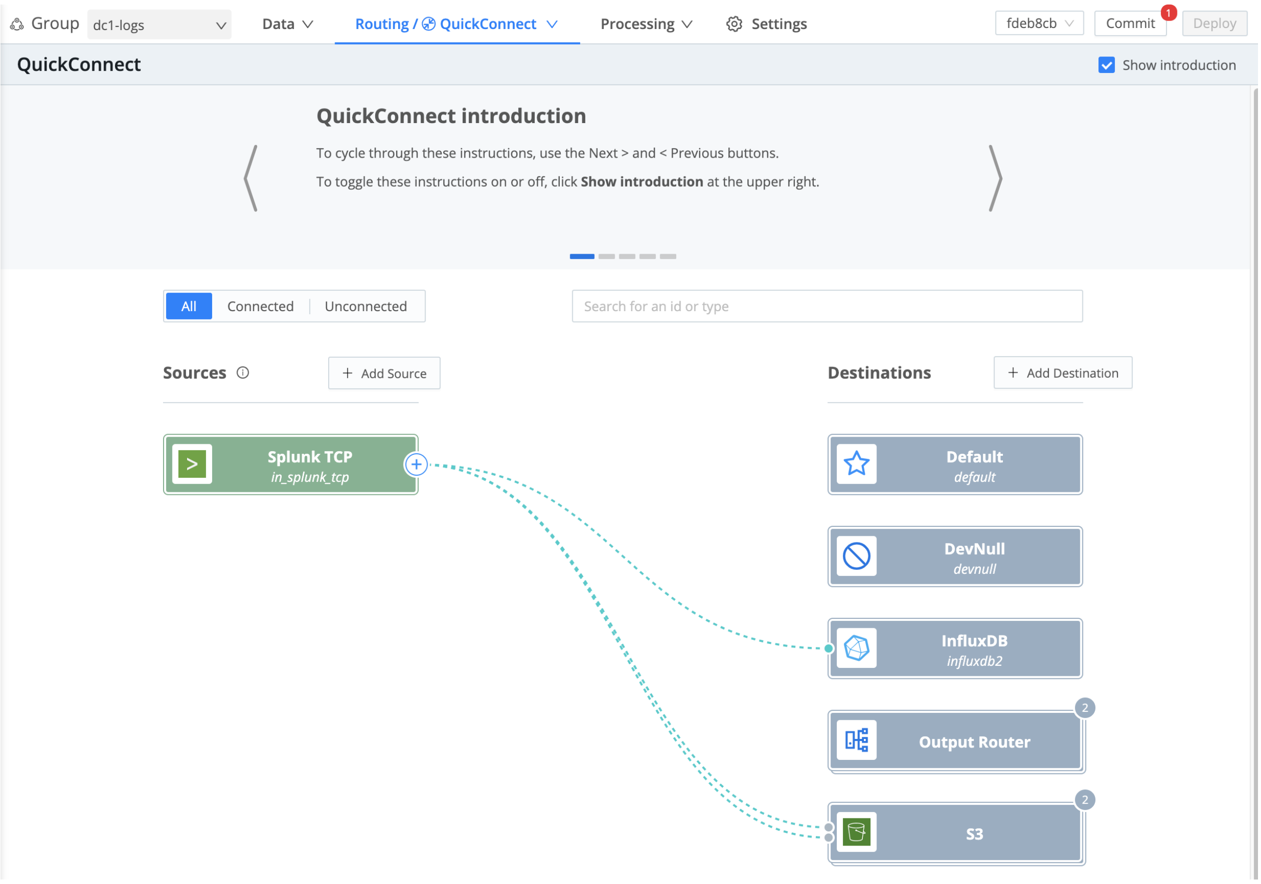
;

1.3. QuickConnect

Cribl Stream 3.2 introduces the QuickConnect visual rapid-development UI. Here, you can 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 introduction

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

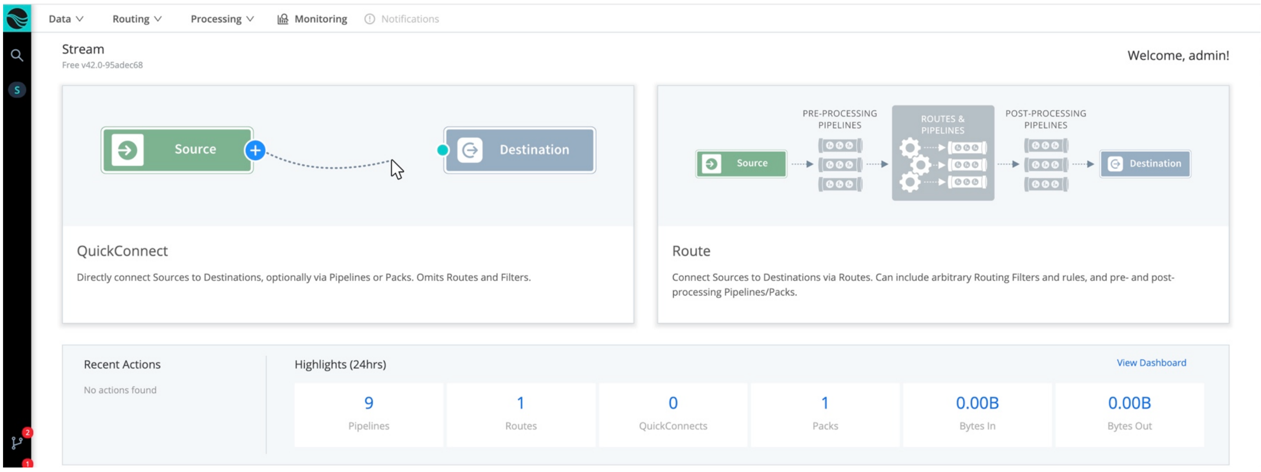
To toggle these instructions on or off, click **Show introduction** at the upper right.

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 (LogStream) 3.2+ 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 or Packs. Omits Routes and Filters.

Route

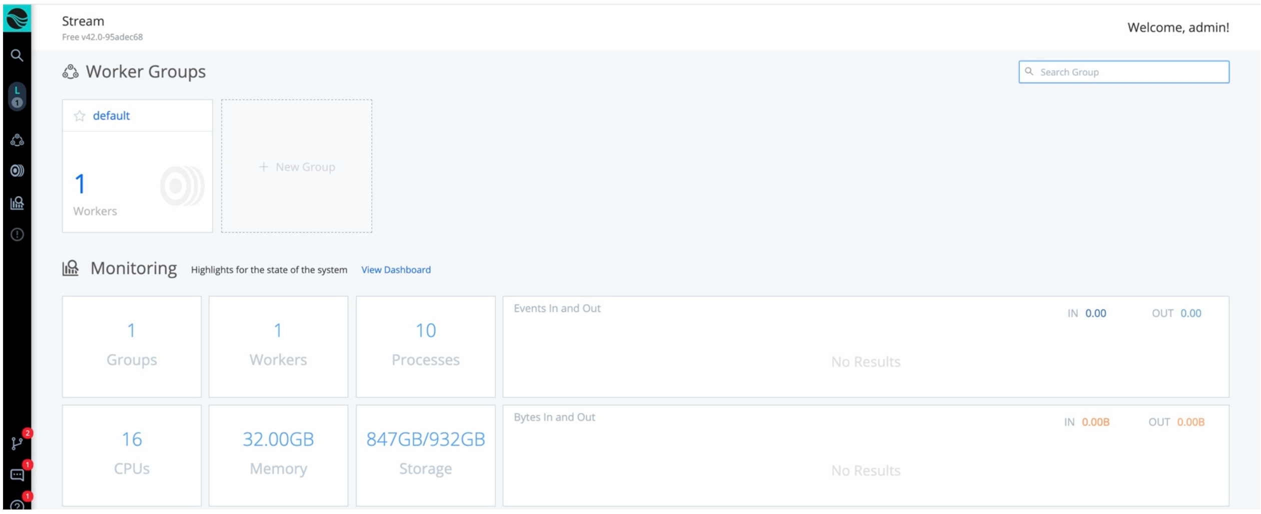
Connect Sources to Destinations via Routes. Can include arbitrary Routing Filters and rules, and pre- and post-processing Pipelines/Packs.

No actions found

9	1	0	1	0.00B	0.00B
Pipelines	Routes	QuickConnects	Packs	Bytes In	Bytes Out

QuickConnect versus Routing UIs

To display the above landing page in a distributed deployment, you must first select a Group. Click the left nav's **Configure** or **Groups** link to select the `default` or a different Group; or click a Group's tile on the distributed landing page, as shown below.



1	1	10
Groups	Workers	Processes
16	32.00GB	847GB/932GB
CPUs	Memory	Storage

default: 1 Workers

Events In and Out: IN 0.00 OUT 0.00

Bytes In and Out: IN 0.00B OUT 0.00B

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**.

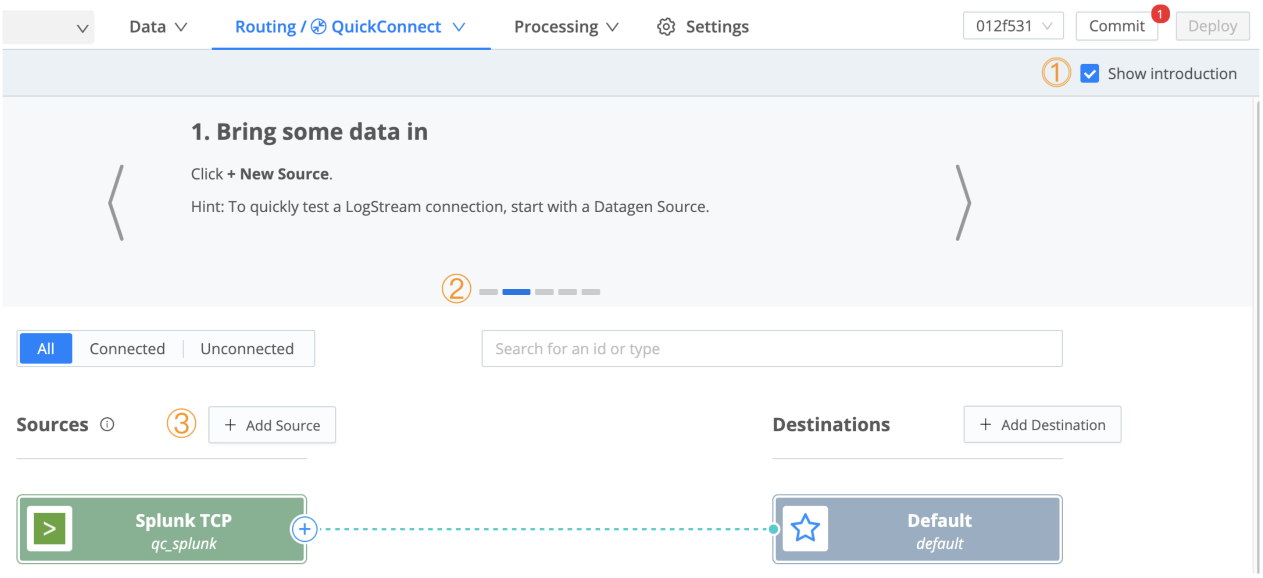
QuickConnect Versus Routes

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:



1. Bring some data in

Click + **New Source**.

Hint: To quickly test a LogStream connection, start with a Datagen Source.

QuickConnect Introduction/help carousel

1. Use the **Show introduction** check box at the upper right to toggle the help text on and off.

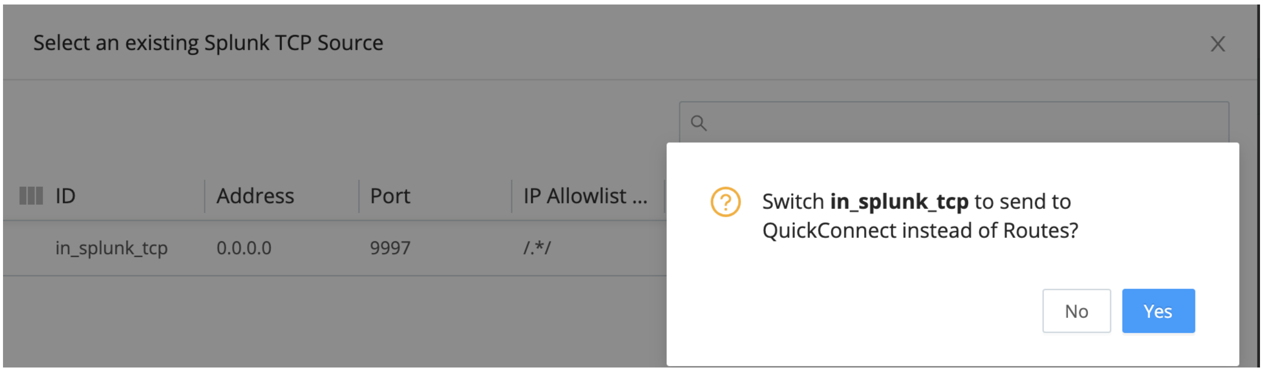
2. Use the dashed buttons to advance or rewind the help text's carousel (whose position is independent of your connection state).
3. Once you've added at least one Source, the **+ New Source** button (mentioned in the help text) becomes an **+ Add Source** button beside the **Sources** header.

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.



Select an existing Splunk TCP Source

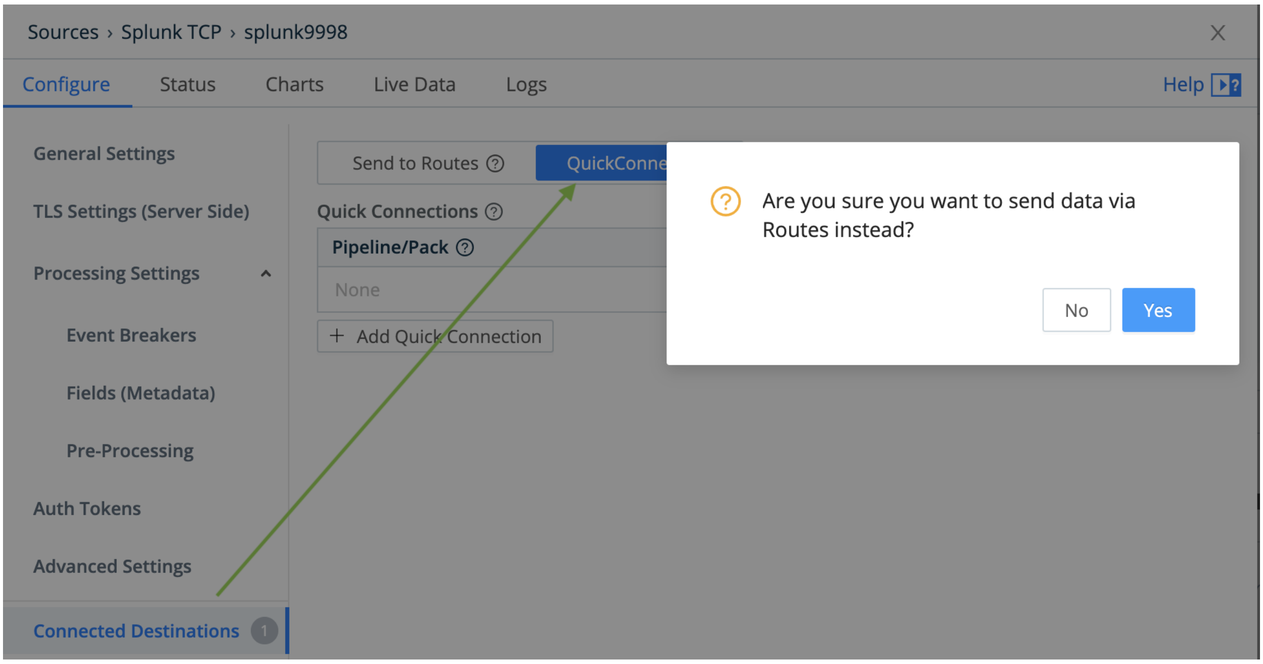
ID	Address	Port	IP Allowlist ...
in_splunk_tcp	0.0.0.0	9997	/.*/

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.

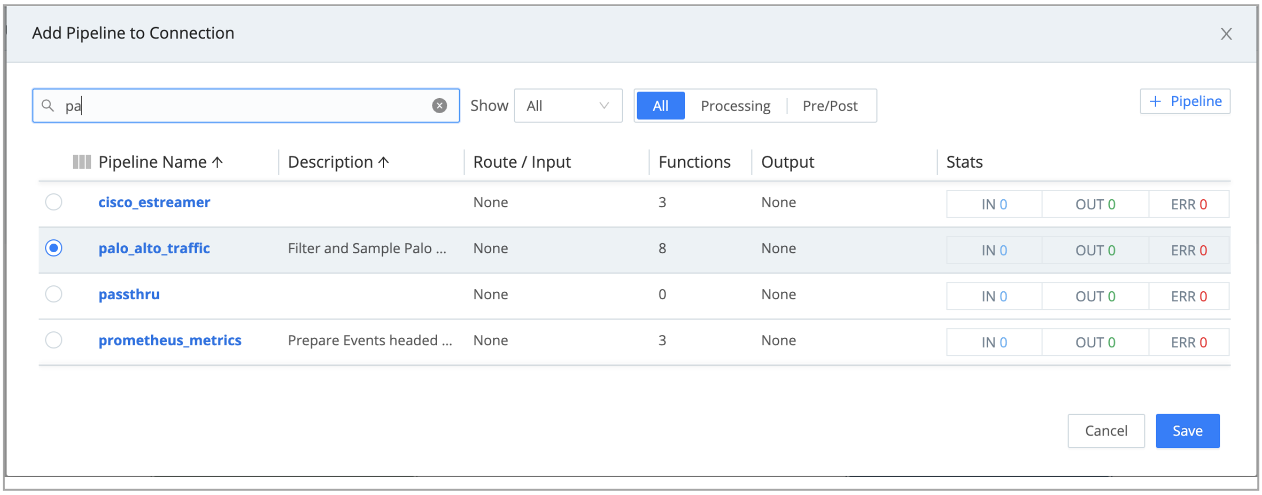


Are you sure you want to send data via Routes instead?

Switching a Source/Destination between worlds

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 Name ↑	Description ↑	Route / Input	Functions	Output	Stats		
○	cisco_estreamer		None	3	None	IN 0	OUT 0	ERR 0
●	palo_alto_traffic	Filter and Sample Palo ...	None	8	None	IN 0	OUT 0	ERR 0
○	passthru		None	0	None	IN 0	OUT 0	ERR 0
○	prometheus_metrics	Prepare Events headed ...	None	3	None	IN 0	OUT 0	ERR 0

Adding a Pipeline

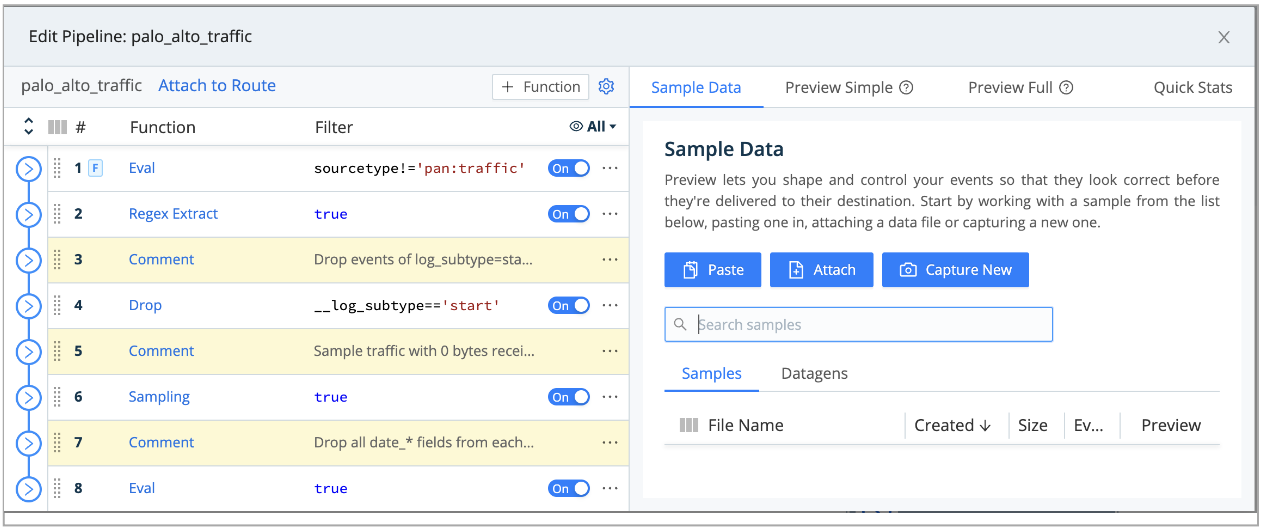
Here, you're using the radio buttons at left to select among already-configured Pipelines or Packs. **Resist the temptation to click on a bold blue Pipeline or Pack name.** (Doing so will open the Pipeline's or Pack's config in a new modal, which might offer more configuration than you bargained for.)

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.



#	Function	Filter
1	Eval	sourcetype!='pan:traffic'
2	Regex Extract	true
3	Comment	Drop events of log_subtype=sta...
4	Drop	__log_subtype=='start'
5	Comment	Sample traffic with 0 bytes recei...
6	Sampling	true
7	Comment	Drop all date_* fields from each...
8	Eval	true

Sample Data

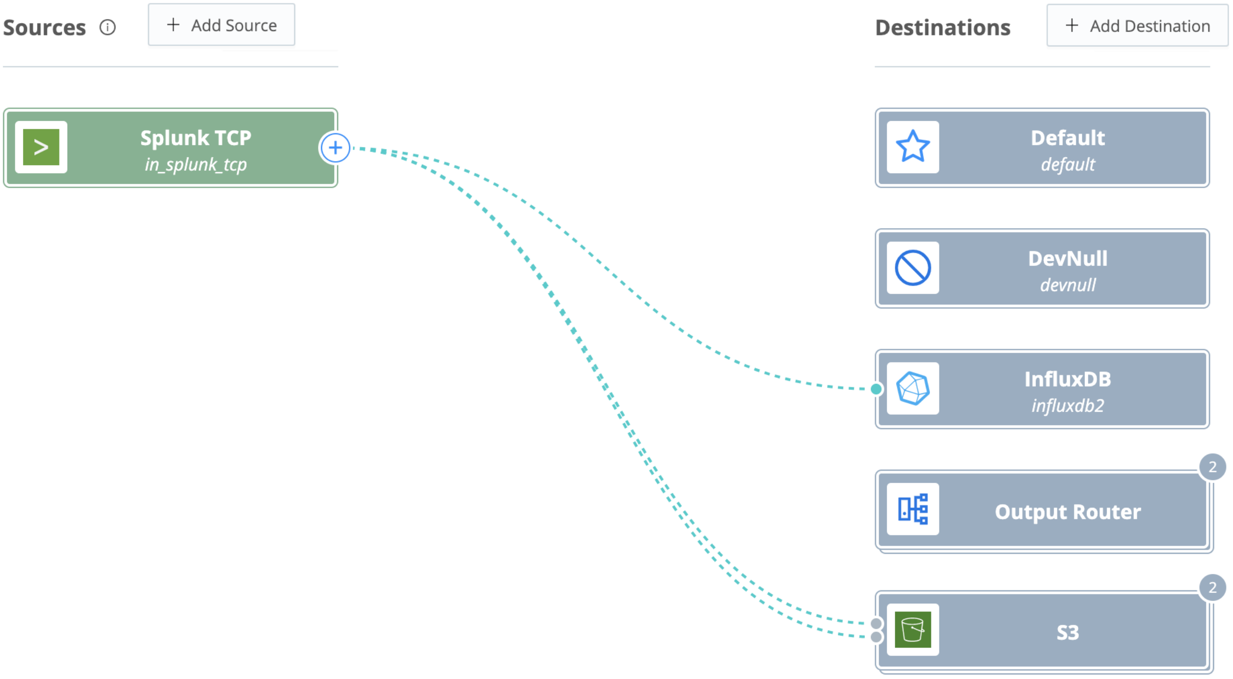
Preview lets you shape and control your events so that they look correct before they're delivered to their destination. Start by working with a sample from the list below, pasting one in, attaching a data file or capturing a new one.

Editing a Pipeline

Multiple Connections

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

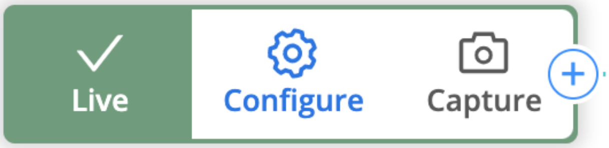
Also, you can connect one Source to multiple Destinations, or multiple Sources to one Destination. Let's isolate examples of both, from the screen capture we started with:



Multiple connections out, multiple connections in

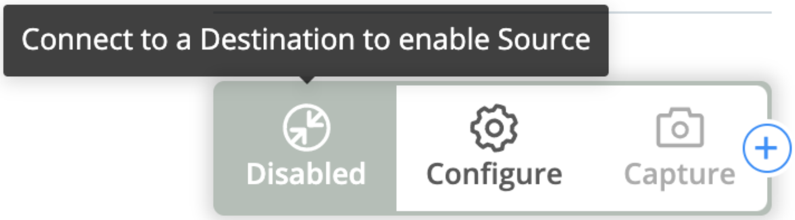
Source/Destination Tiles' Hover Options

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



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

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



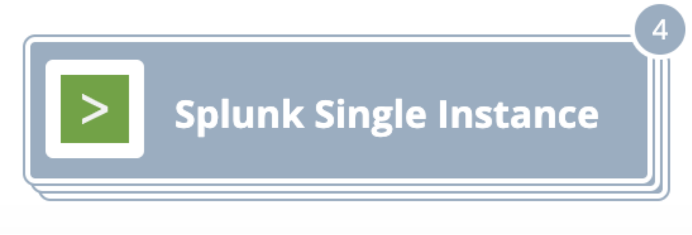
This Source is not misconfigured – it's just lonely

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

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

Group Connections

Once you've configured two or more copies of the same Source or Destination type – for example, to support different configurations of one integration or protocol – QuickConnect stacks their tiles together to keep the display clean.



4 similar Destinations, stacked

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

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

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

;

1.4. 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 Cloud or self-hosted), skip ahead to Get Data Flowing.
>
> Once you've mastered all the techniques in this tutorial, check out its Distributed Quick Start successor.

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 (Intel Processors)

- Linux 64-bit kernel >= 3.10 and glibc >= 2.17
- Examples: Ubuntu 16.04, Debian 9, RHEL 7, CentOS 7, SUSE Linux Enterprise Server 12+, Amazon Linux 2014.03+

OS (ARM64 Processors)

- Linux 64-bit
- Tested so far on Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2

System

- +4 physical cores, +8GB RAM – all beyond your basic OS/VM requirements

- 5GB 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 Support

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

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!

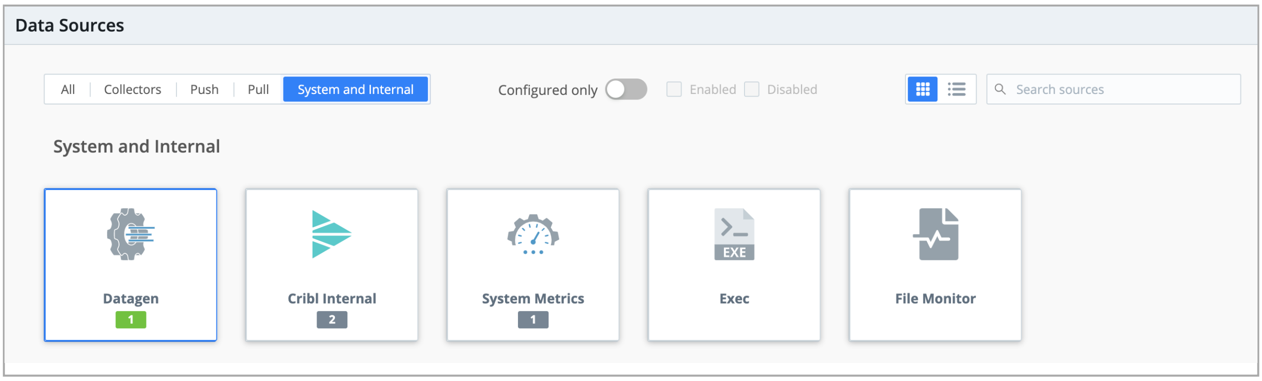
Get Data Flowing

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 left nav's **Configure** or **Groups** link to select the `default` (or another) Worker Group.

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

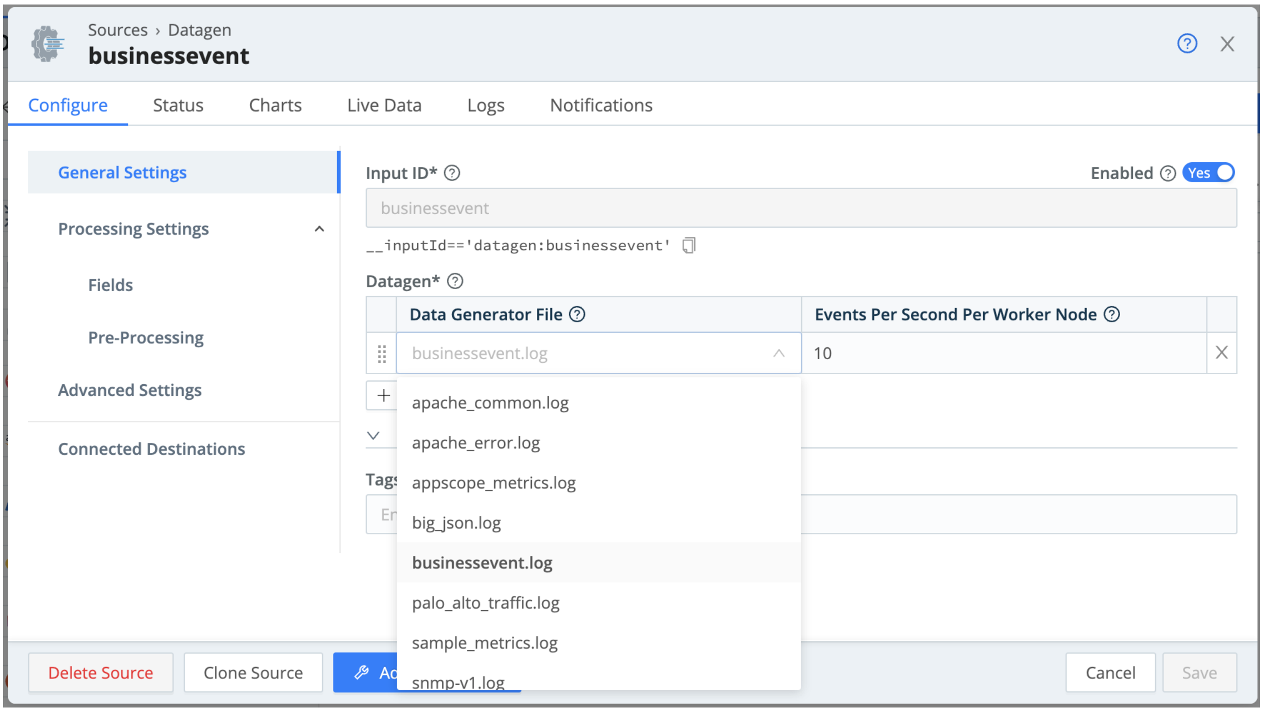
   (You can use the search box to jump to the **Datagen** tile.)
3. Click **+ Add New** to open the **New Datagen source** pane.
4. In the **Input ID** field, name this Source `businessevent`.
5. In the **Data Generator File** drop-down, select `businessevent.log`.

   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** slider in the **Enabled** column indicates that your Datagen Source has started generating sample data.


Sources › Datagen
businessevent

Input ID*: businessevent
Enabled: Yes
`__inputId=='datagen:businessevent'`

Datagen*

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

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 left nav's **Configure** or **Groups** link 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.

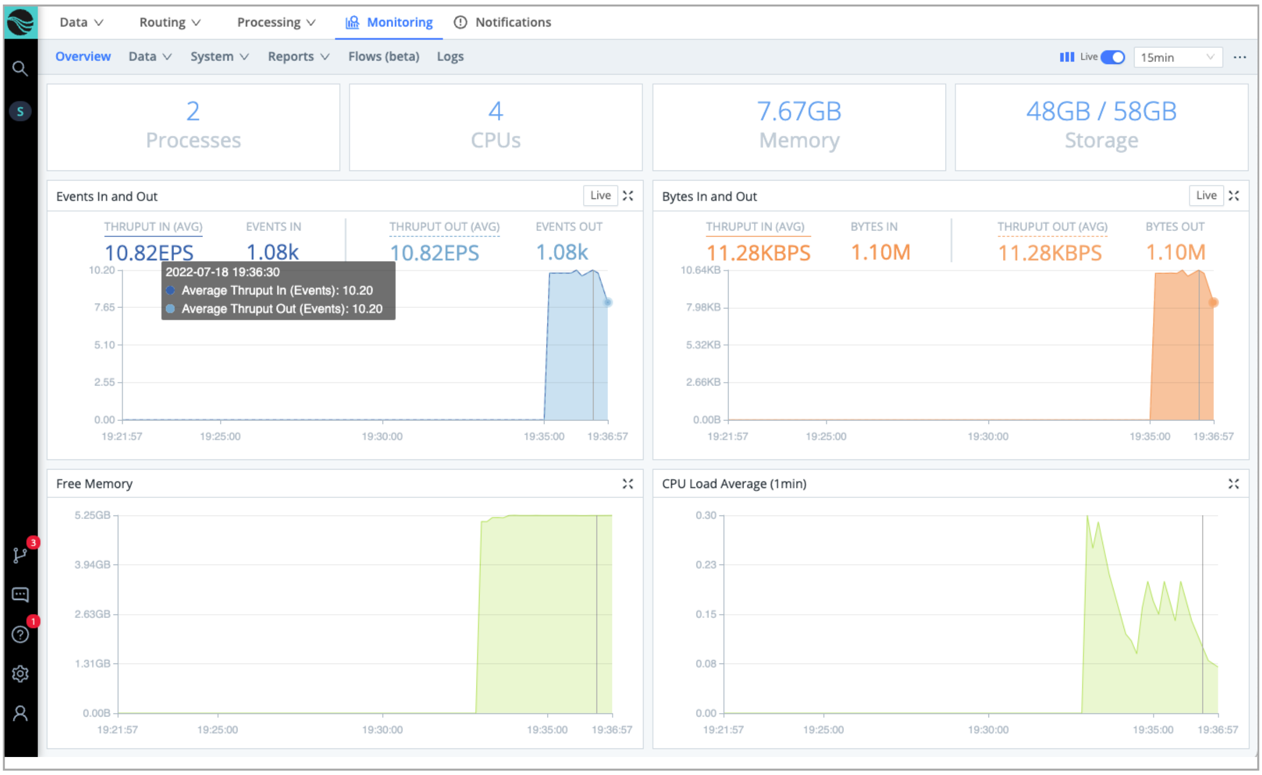
> If you're on Cribl.Cloud or any other distributed mode, click **Commit & Deploy** at Cribl Stream's upper right (if available) before proceeding. Then, in the resulting dialog box, click **Commit & Deploy** to confirm. You'll see a **Commit successful** message.

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

Monitor Data Throughput

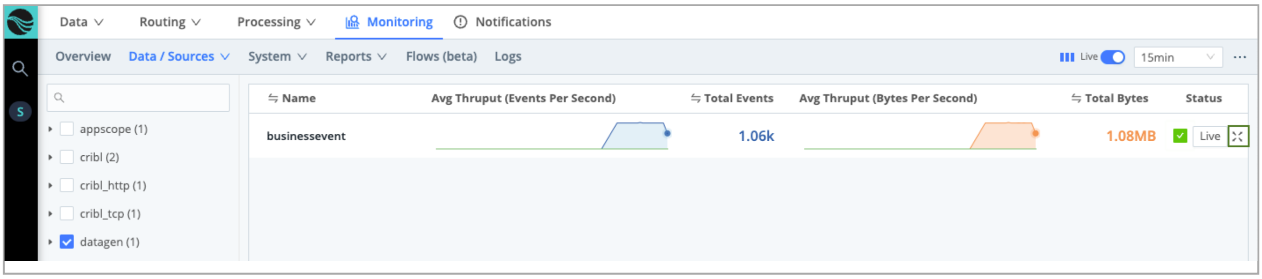
On a single-instance deployment, click the top nav's **Monitoring** link. (On very narrow displays, you might need to select it from the ••• overflow menu.) On Cribl.Cloud or any other distributed mode, click the left nav's **Monitoring** link.

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:

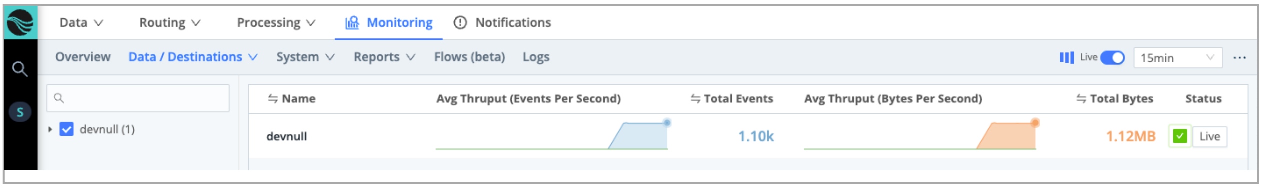


Name	Avg Thruput (Events Per Second)	Total Events	Avg Thruput (Bytes Per Second)	Total Bytes	Status
businessevent		1.06k		1.08MB	✓ Live

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 (Events Per Second)	Total Events	Avg Thruput (Bytes Per Second)	Total Bytes	Status
devnull		1.10k		1.12MB	✓ Live

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 left nav's **Configure** or **Groups** link to select the `default` (or another) Worker Group.

1. From the top menu, select **Processing** > **Pipelines**.

   You now have a two-pane view, with ~~business on the left and party on the right~~ 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 **+ Pipeline**, then select **Create Pipeline**.

3. In the new Pipeline's **ID** field, enter a unique identifier. (For this tutorial, you might use `slicendice`.)

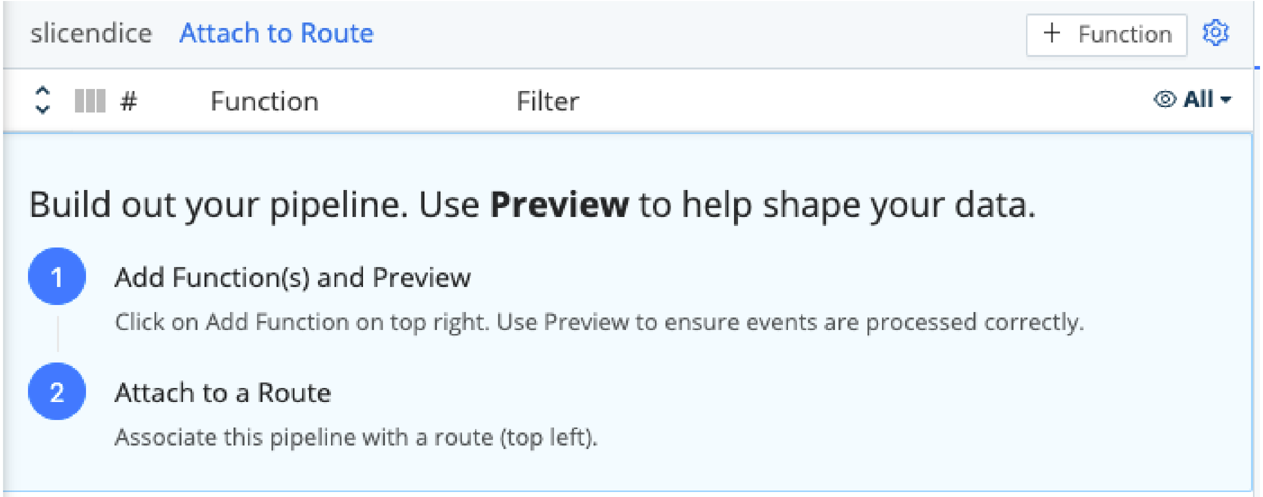
4. Optionally, enter a **Description** of this Pipeline's purpose.

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.


slicendice Attach to Route

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 New**.
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 **Preview Simple** 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 **+ Function**.

2. Search for `Mask`, then click it.

3. In the new Function's **Masking Rules**, click the into **Match Regex** field.

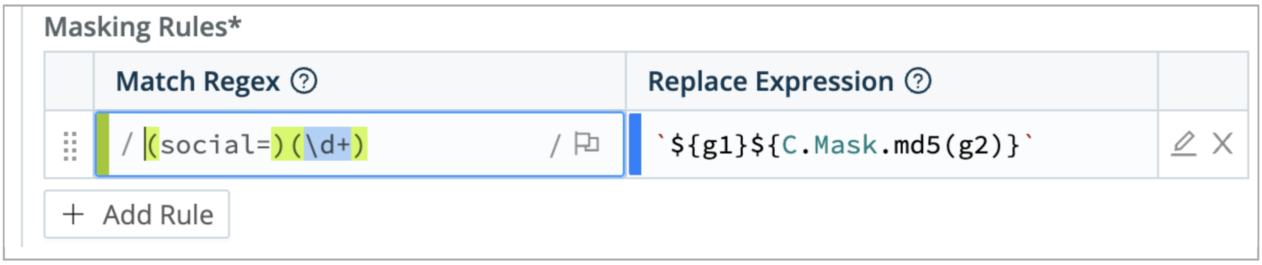
   Now we want to build the **Match Regex**/**Replace Expression** row shown just below.

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*

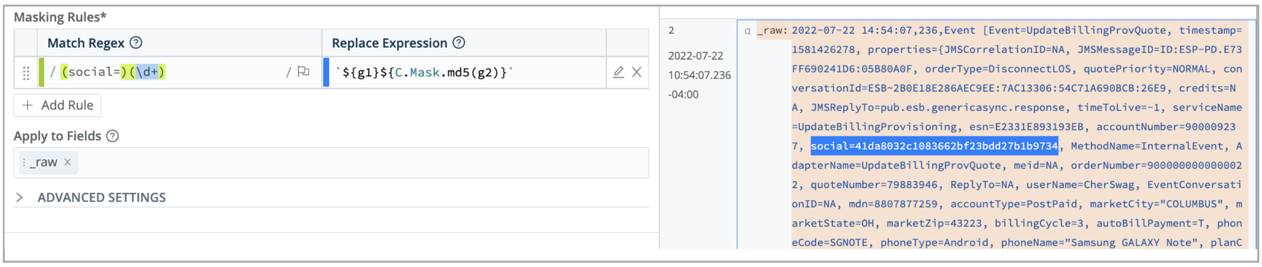
Match Regex	Replace Expression
`/(social=)(\d+)/`	`` `${g1}${C.Mask.md5(g2)}` ``

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

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

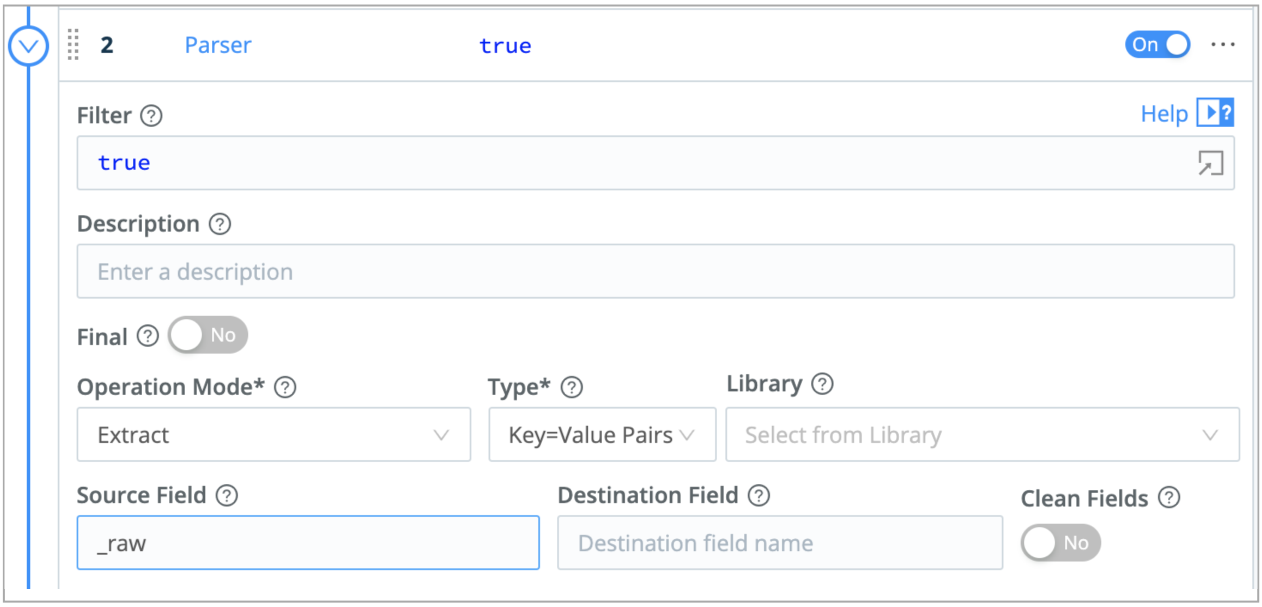
```
_raw: 2022-07-22 14:54:07,236,Event [Event=UpdateBillingProvQuote, timestamp=
      1581426278, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.E73
      FF690241D6:05B80A0F, orderType=DisconnectLOS, quotePriority=NORMAL, con
      versationId=ESB~2B0E18E286AEC9EE:7AC13306:54C71A690BCB:26E9, 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 **+ 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**.



2 Parser true On

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

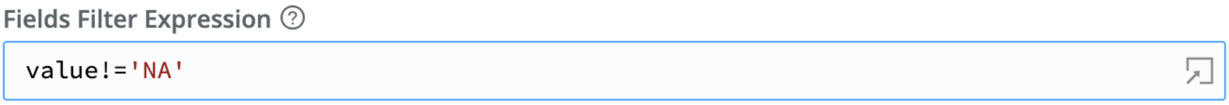
Parser configured to extract K=V pairs from `_raw`

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.



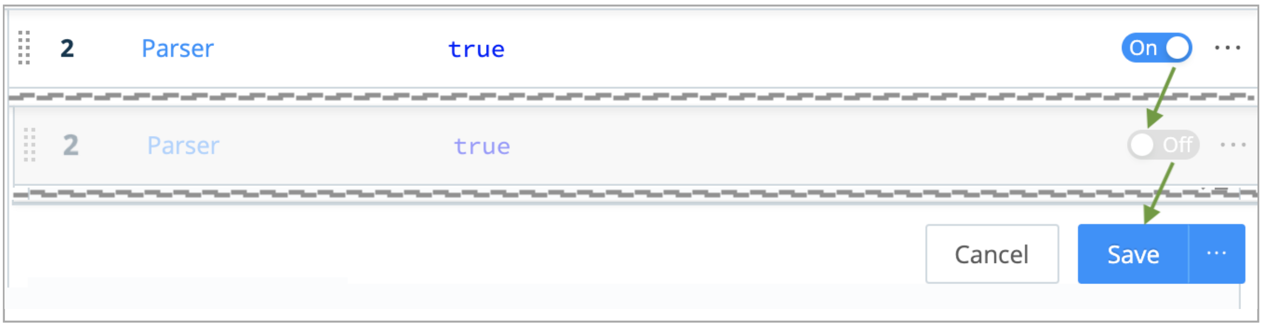
Fields Filter Expression: value!='NA'

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, slide 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 **+ Function**.

2. Search for `Eval`, then click it.

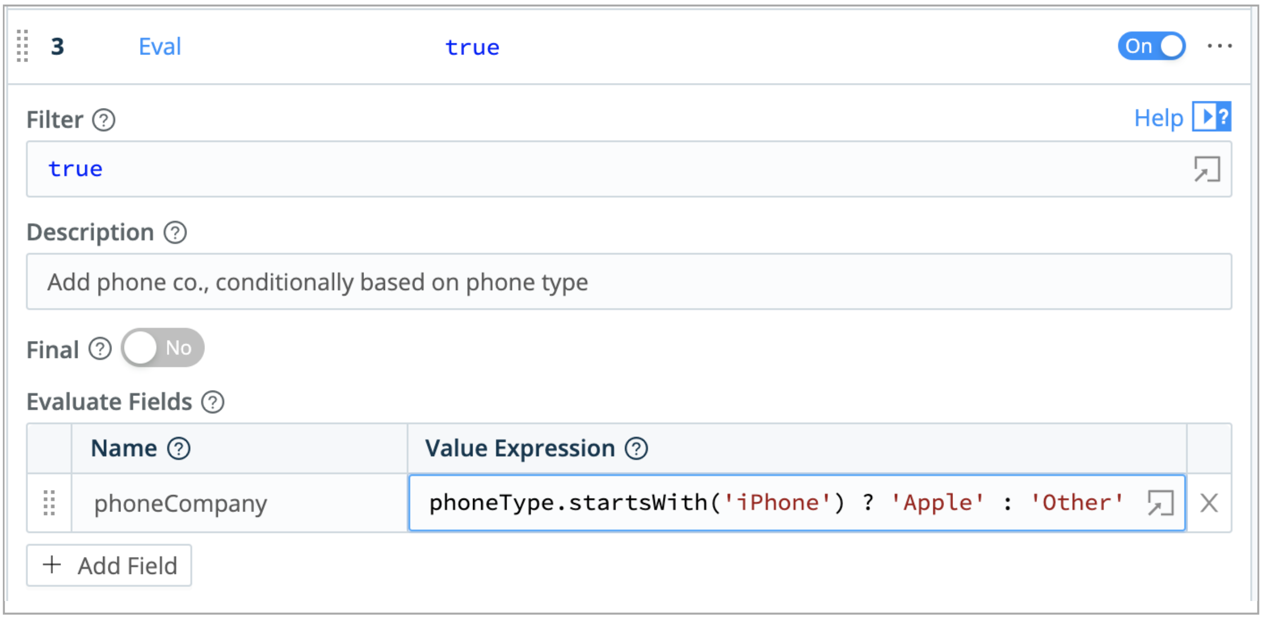
3. Click **+ Add Fields** to open the **Evaluate Fields** table.

   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`.



Adding a field to enrich data

Remove Field (Shrink Data)

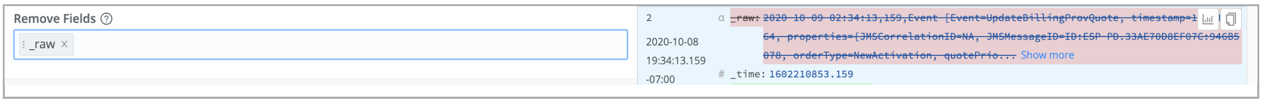
Now that we've parsed out all of the `_raw` field's data – it can go. Deleting a (large) redundant field will give us cleaner events, and reduced load on downstream resources.

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.



```
~~_raw: 2020-10-09 02:34:13,159,Event [Event=UpdateBillingProvQuote, timestamp=1
64, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.33AE70D8EF07C:946B5
078, orderType=NewActivation, quotePrio...~~
_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 `+ 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 **+ 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'`

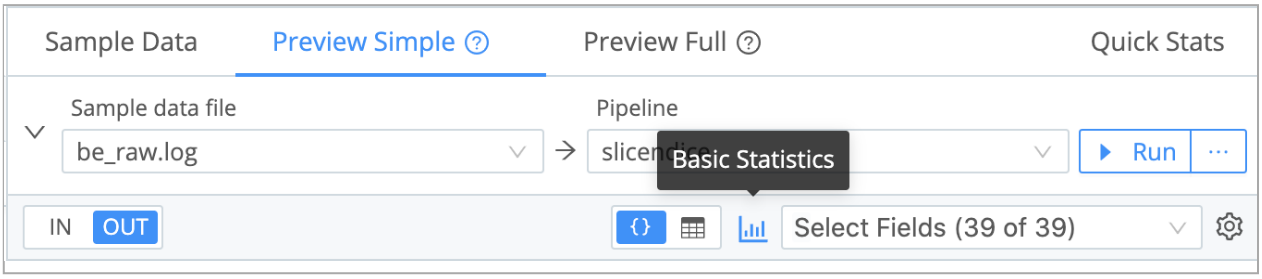
5. Click **Save**.

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

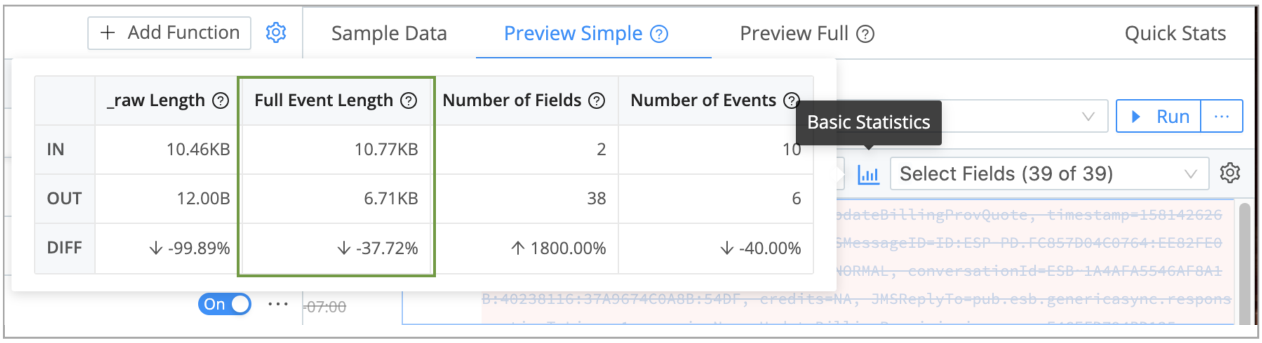
Torture the data enough, and it will confess. By what factor have our transformations refined our data's volume? Let's check.

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%

~~-07:00~~
~~odateBillingProvQuote, timestamp=158142626~~
~~SMessageID=ID:ESP-PD.FC857D04C0764:EE82FE0~~
~~NORMAL, conversationId=ESB-1A4AFA5546AF8A1~~
~~B:40238116:37A9674C0A8B:54DF, credits=NA, JMSReplyTo=pub.esb.genericasync.respons~~

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 `+ 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`.

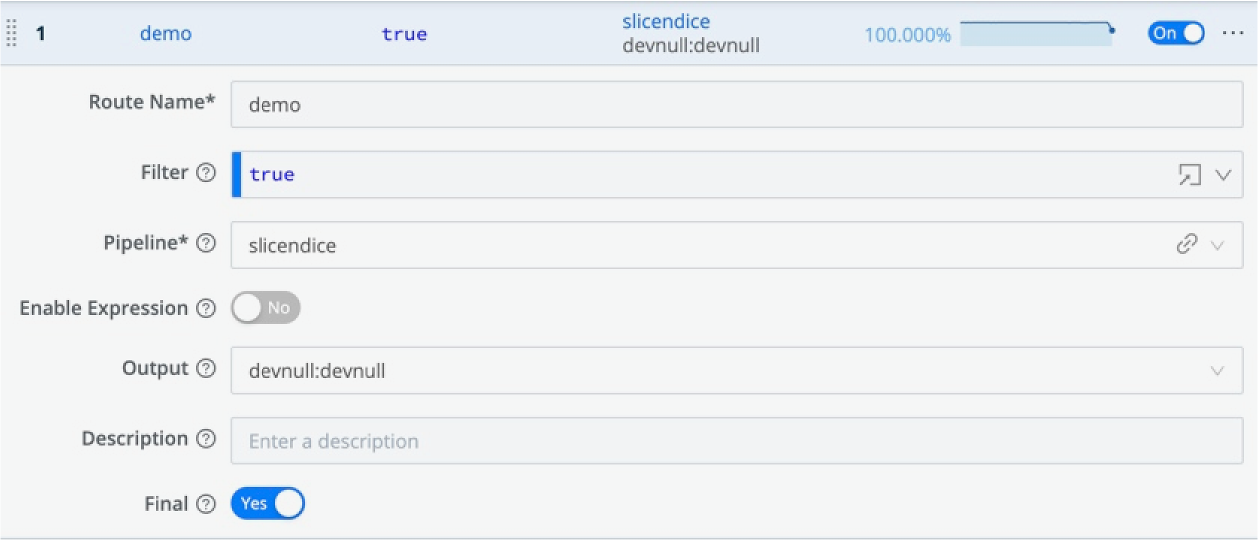
   This doesn't matter, because we've set `default` as a pointer to `devnull`. In production, you'd set this carefully.

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	
1	demo	true	slicendice devnull:devnull	100.000%	On

Route Name*: demo

Filter: true

Pipeline*: slicendice

Enable Expression: No

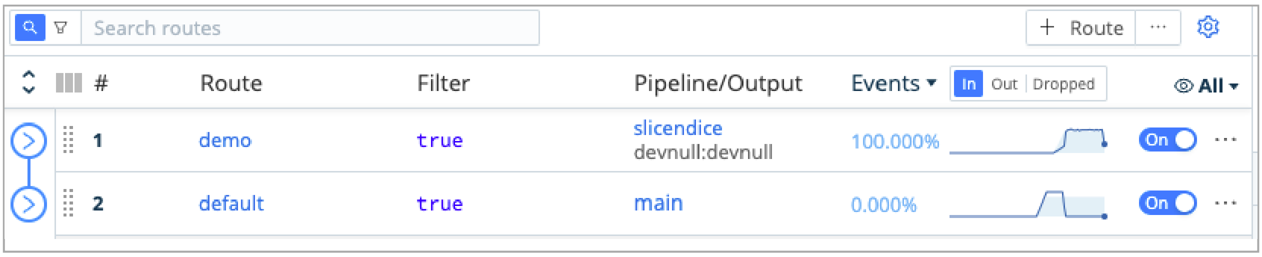
Output: devnull:devnull

Description: Enter a description

Final: Yes

Configuring and adding a Route

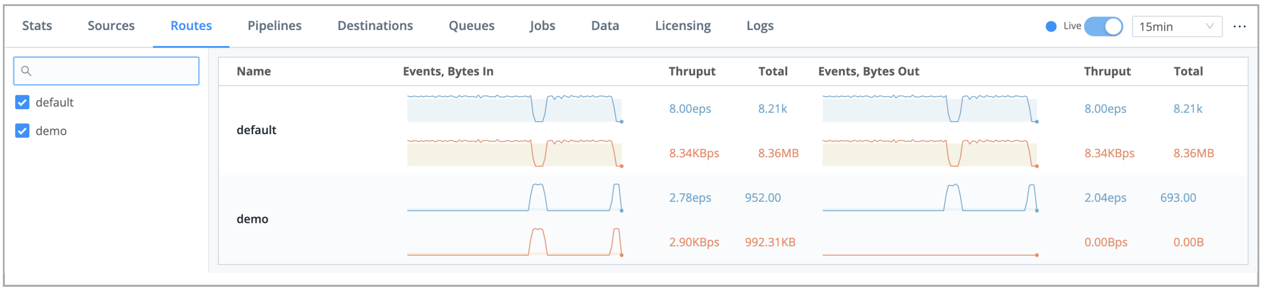
The sparklines should immediately confirm that data is flowing through your new Route:



#	Route	Filter	Pipeline/Output	Events	
1	demo	true	slicendice devnull:devnull	100.000%	On
2	default	true	main	0.000%	On

Live Routes

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



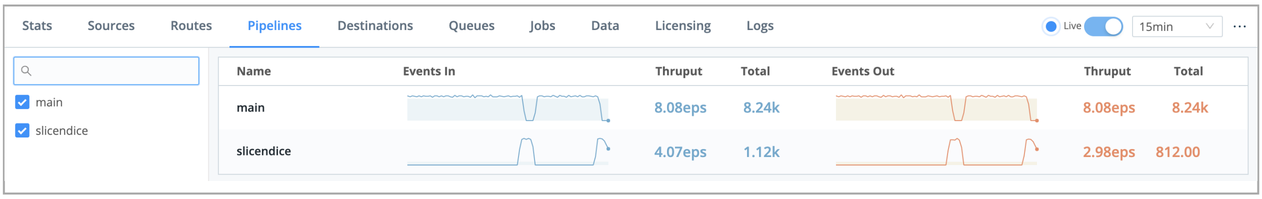
☑ default

☑ demo

Name	Thruput	Total	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`.



☑ main
☑ 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

Monitoring data flow through Pipelines

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. Slide `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.

;

1.5. 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.

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. 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, 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 this `docker-compose.yml` to easily stand up a Cribl Stream distributed deployment of a Leader and multiple Workers on one machine:

```
version: '3.5'
services:
  master:
    image: ${CRIBL_IMAGE:-cribl/cribl:latest}
    environment:
      - CRIBL_DIST_MODE=master
      - CRIBL_DIST_MASTER_URL=tcp://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_IMAGE:-cribl/cribl:latest}
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://criblmaster@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          STATUS
PORTS                                              NAMES
a3de9ea8f46f   cribl/cribl:latest    "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
seconds   0.0.0.0:63411->9000/tcp                       docker_workers_1
40aa687baefc   cribl/cribl:latest    "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
seconds   0.0.0.0:63410->9000/tcp                       docker_workers_2
df362a65f7d1   cribl/cribl:latest    "/sbin/entrypoint.sh…"   13 seconds ago   Up 11
seconds   0.0.0.0:19000->9000/tcp, :::19000->9000/tcp   docker_master_1
```

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 zxvf -`

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

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=criblmaster&tag=<tag1>&tag=<tag2>'
```

E.g.: `curl 'http://localhost:9000/init/install-worker.sh?token=criblmaster&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 for EC2 deployment:
    - Tutorial: Getting started with Amazon EC2 Linux instances
    - Launching an instance using the Launch Instance Wizard
    - How can I create and connect to an Amazon Linux EC2 instance?
2. See these 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.

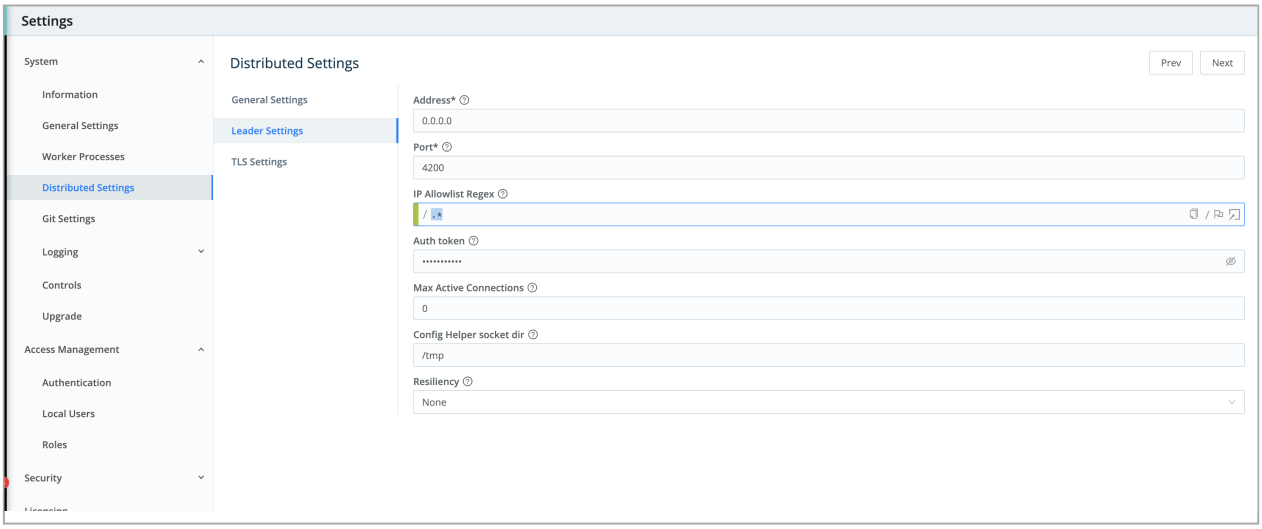
3. From the UI's global ⚙ **Settings** (lower left) > **Distributed Settings** > **Distributed Management** > **General Settings**, select **Mode**: `Leader`.

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. Optional but recommended: Enable the nearby **Remote UI access** slider. This way, you will be able to click through from the Leader's **Manage Workers** page to view and manage each Worker's UI (distinguished by an orange header). This option handles authentication for you, so you don't need to manage or enter the Workers' credentials.

6. Click **Save** to restart.

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



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

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 global ⚙ **Settings** (lower left) > **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 `criblmaster`.)

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-
token> | [sudo] sh -
```

Or, for bash:

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

> 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 left nav, click **Manage**.

3. From the resulting fly-out or **Manage Groups** page, 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 **Remote 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 left nav, click **Manage**.

2. From the resulting fly-out or **Manage 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 **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 New** to open the **New Datagen source** pane.

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.

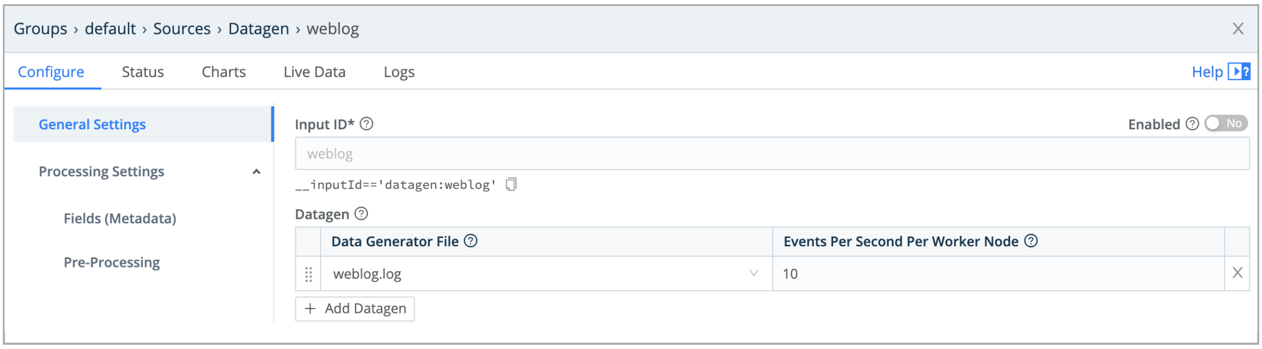
8. Keep the **Events per Second per Worker Node** at the default `10` EPS for now.

9. Click **Save**.

   The **On** slider in the **Enabled** column indicates that your Datagen Source has started generating sample data.

10. Click **Commit** at the upper right, enter a commit message, and confirm the commit.
    This uses Cribl Stream's `git` integration to record the `default` Group's new configuration.

11. Click **Deploy** at the upper right to deploy this new configuration to your Workers.



Groups › default › Sources › Datagen › weblog

Input ID*: weblog

`__inputId=='datagen:weblog'`

Enabled: No

Datagen

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

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 Datagen Sources** 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

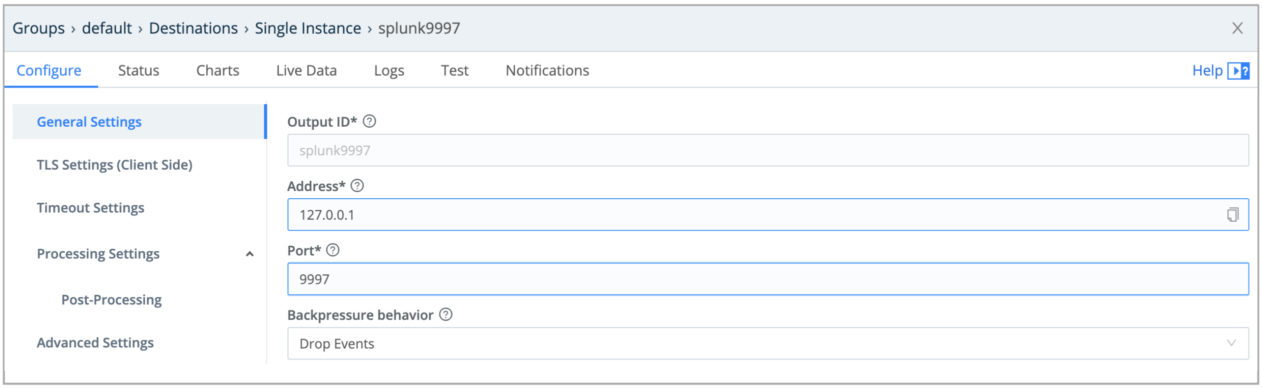
On the output side, choose any of these options:

- 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`.



Output ID*: splunk9997
Address*: 127.0.0.1
Port*: 9997
Backpressure behavior: Drop Events

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. 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 global ⚙ **Settings** (lower left) > **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 left nav, click **Manage**.

2. From the resulting fly-out or **Manage 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 **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.

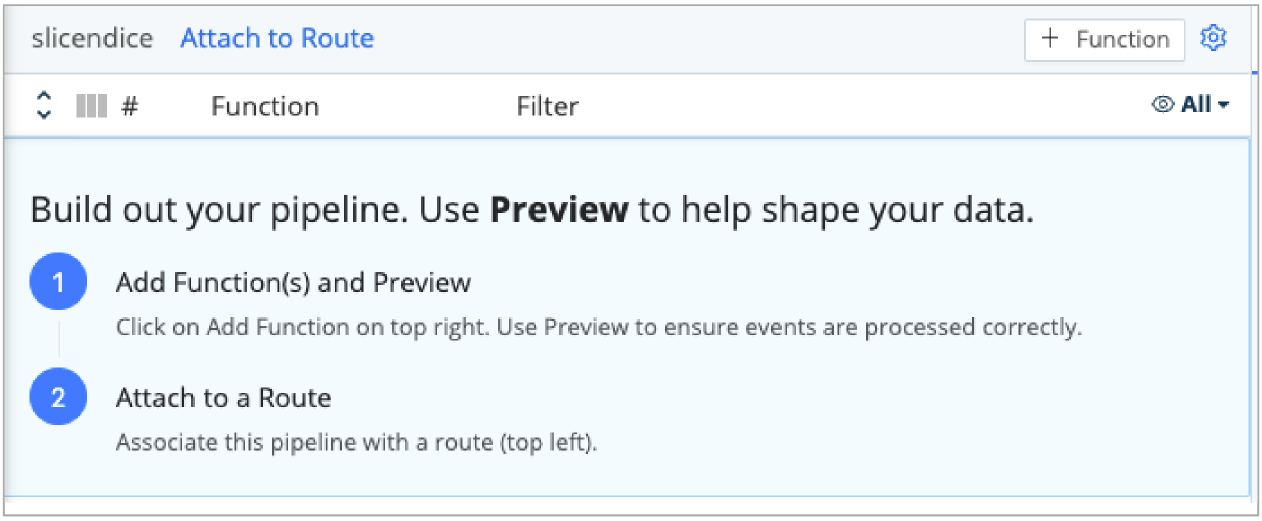
1. From the Leader's top menu, select **Processing** > **Pipelines**.

2. At the **Pipelines** pane's upper right, click **+ 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).

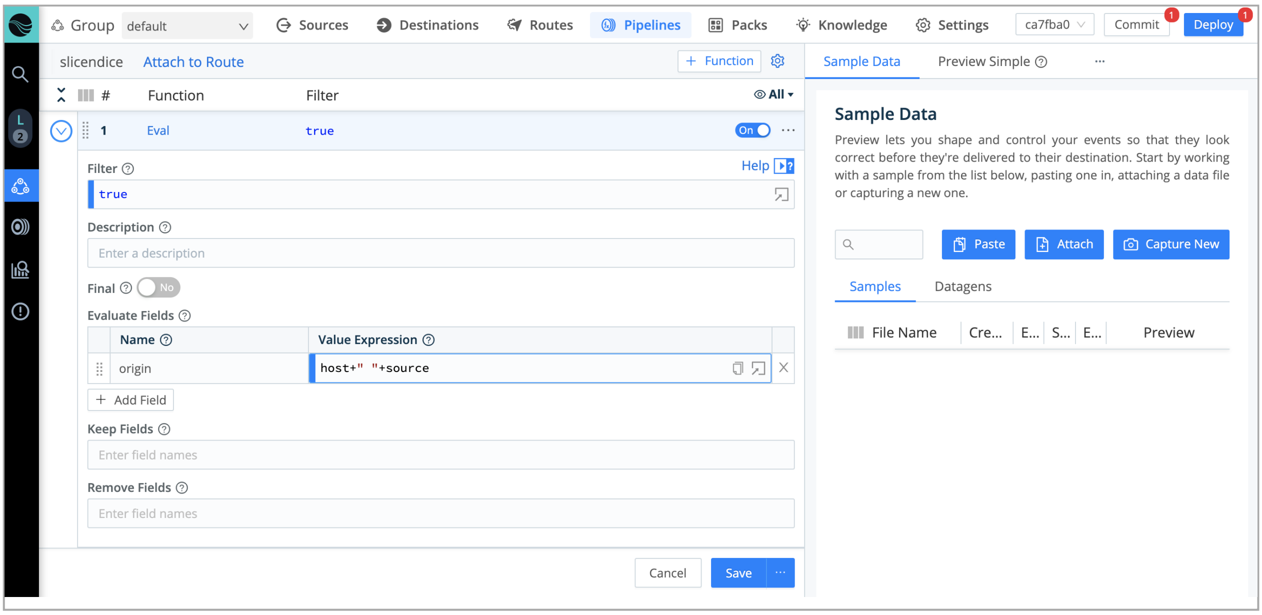
Pipeline saved

Add an Eval Function

Now add an Eval Function to the Pipeline:

1. At the **Pipelines** pane's upper right, click **+ 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.



#	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

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.

Adding Eval Function to Pipeline

6. Click **Save** to store the Function's configuration.
7. **Commit** and **Deploy** the new Pipeline configuration.
8. Optionally, click **Capture New** in the right pane, and capture some 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 **Routes** page.

2. Click `+ 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`.

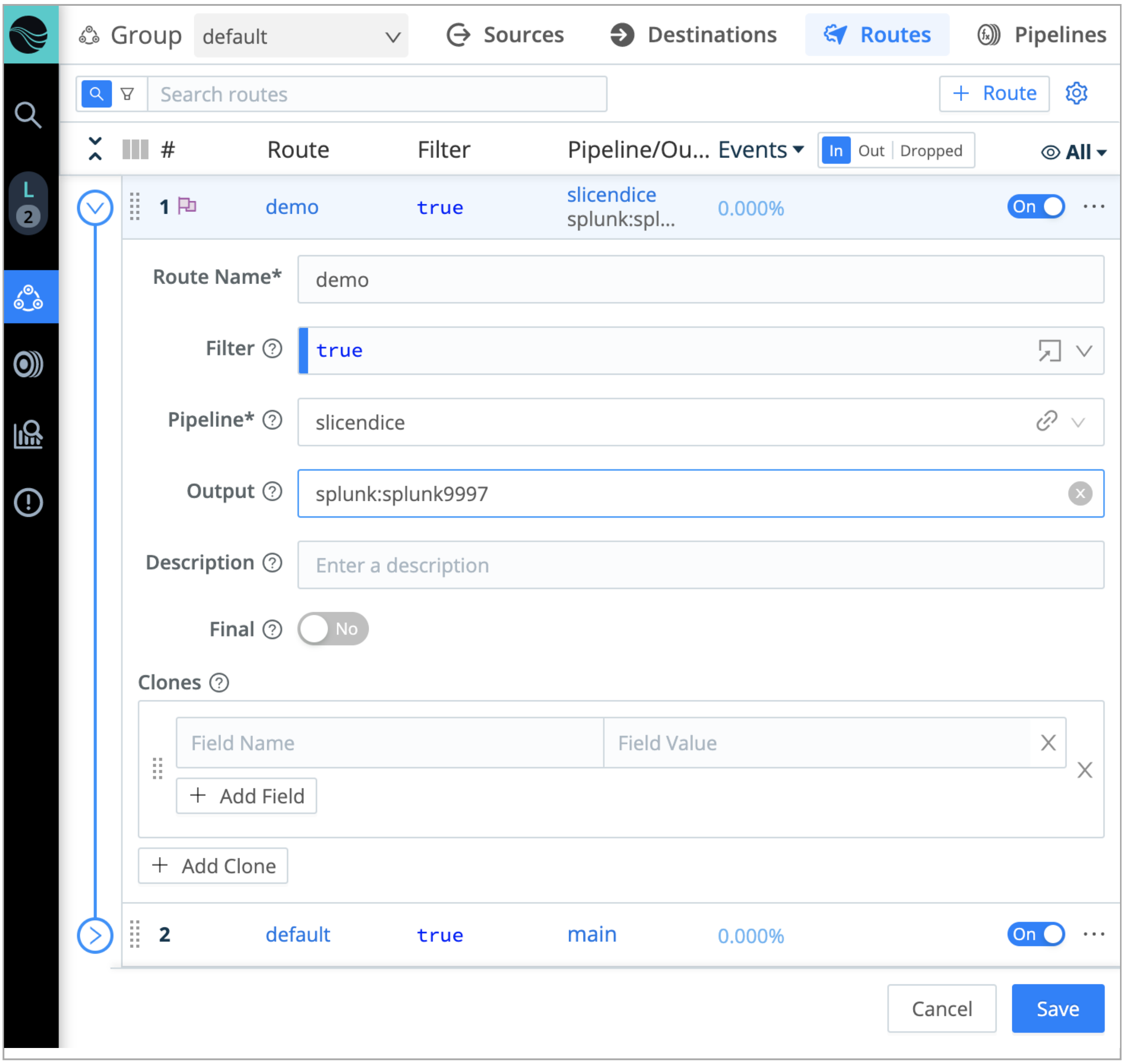
7. You can leave the **Description** empty, and leave **Final** set to **Yes**.

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/Ou...	Events
1	demo	true	slicendice splunk:spl...	0.000%
2	default	true	main	0.000%

Route Name*: demo

Filter: `true`

Pipeline*: slicendice

Output: splunk:splunk9997

Description: Enter a description

Final: No

Clones

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.

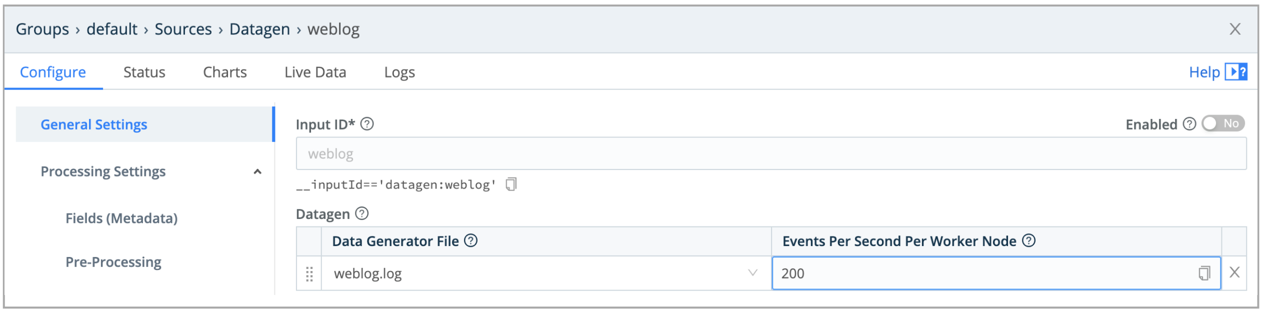
1. In the Leader UI's left nav, click **Manage**.

2. From the resulting fly-out or **Manage 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 **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.



Groups › default › Sources › Datagen › weblog

Configure | Status | Charts | Live Data | Logs | Help

General Settings
Processing Settings
Fields (Metadata)
Pre-Processing

Input ID*: weblog
Enabled: No

`__inputId=='datagen:weblog'`

Datagen

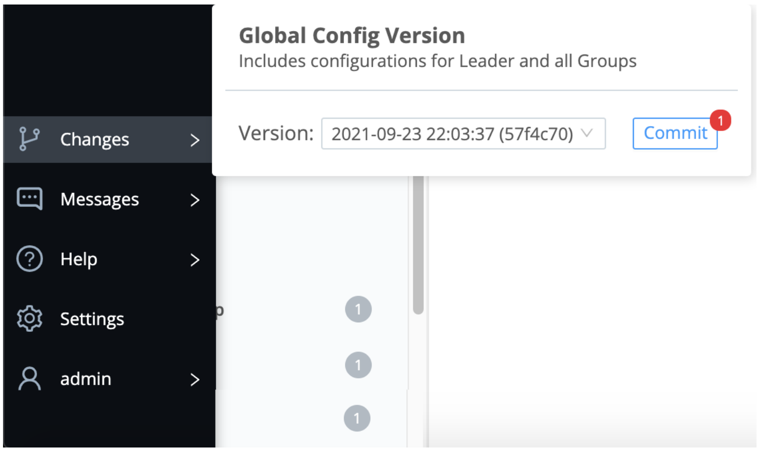
Data Generator File	Events Per Second Per Worker Node
weblog.log	200

Flooding events into Cribl Stream

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

6. From the left nav, also select **Changes** > **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.

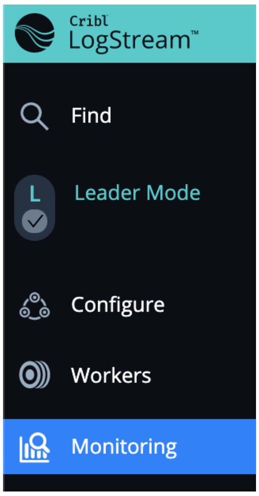


Global Config Version
Includes configurations for Leader and all Groups

Version: 2021-09-23 22:03:37 (57f4c70)

Committing Leader's config

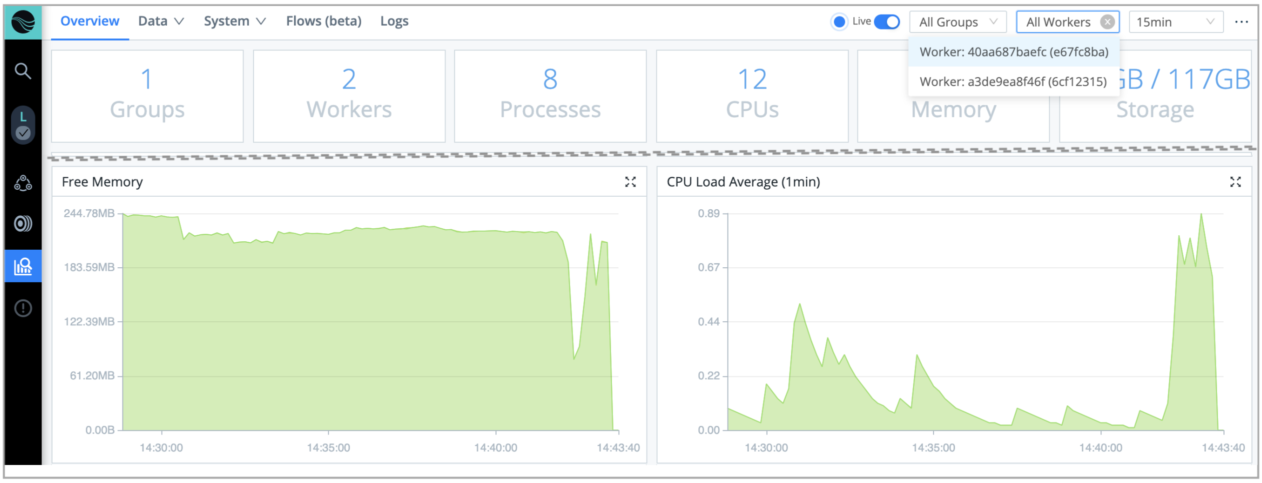
7. From the left nav, click **Monitoring**.



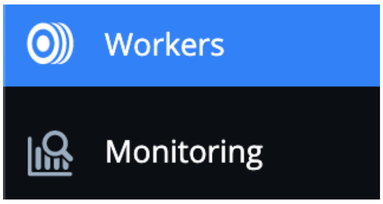
Monitoring tab

8. 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.



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



Workers tab

10. 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 a Cribl Stream Enterprise or Sales Trial license.

> 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 (left nav), before you proceed.

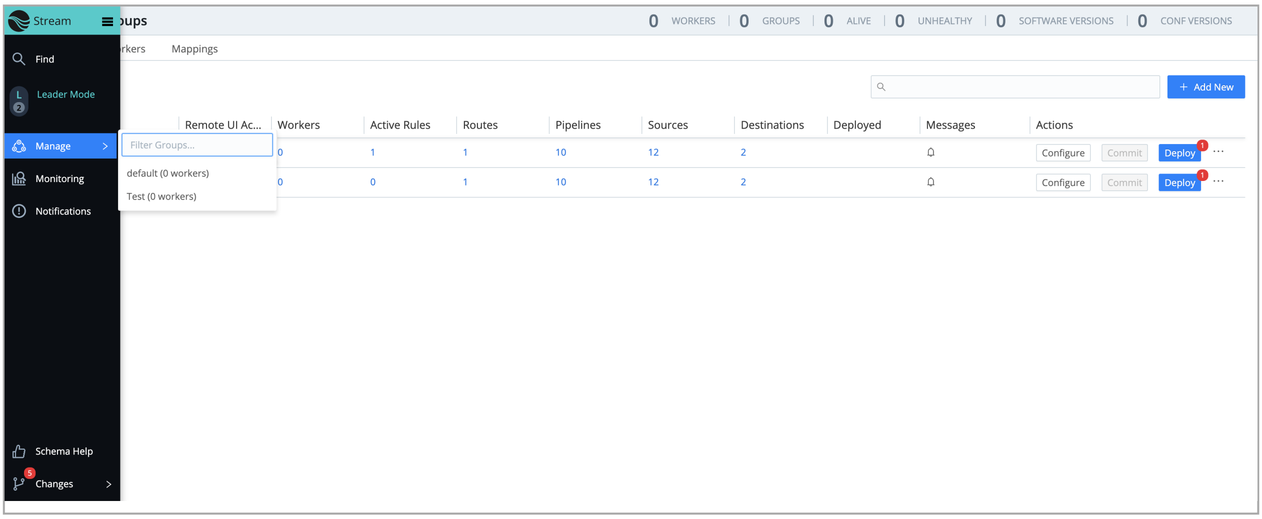
Here, we'll build on the infrastructure we've created so far to:

- 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, Cribl Stream's UI adds some extra features, shown here.



Remote UI Ac...	Workers	Active Rules	Routes	Pipelines	Sources	Destinations	Deployed	Messages	Actions
	0	1	1	10	12	2			Configure Commit Deploy
	0	0	1	10	12	2			Configure Commit Deploy

UI for multiple Worker Groups

Note that:

- The top header now shows you the number of configured Groups (`1`), along with other statistics.
- The left nav's **Groups** option is renamed **Manage**.
- Clicking that option opens a submenu of Worker Groups. It currently offers Cribl Stream's single default Group, literally named `default`.
- If you've enabled Worker UI access, you can use the second-level submenu to click directly through to each of your Workers. (A purple header indicates that you're viewing a Worker's UI. – we might use this feature just below. To return to the Leader's UI, just click the left nav's **Manage** option.)

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 left nav's **Workers** tab, 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 **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 left nav's **Groups** 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.

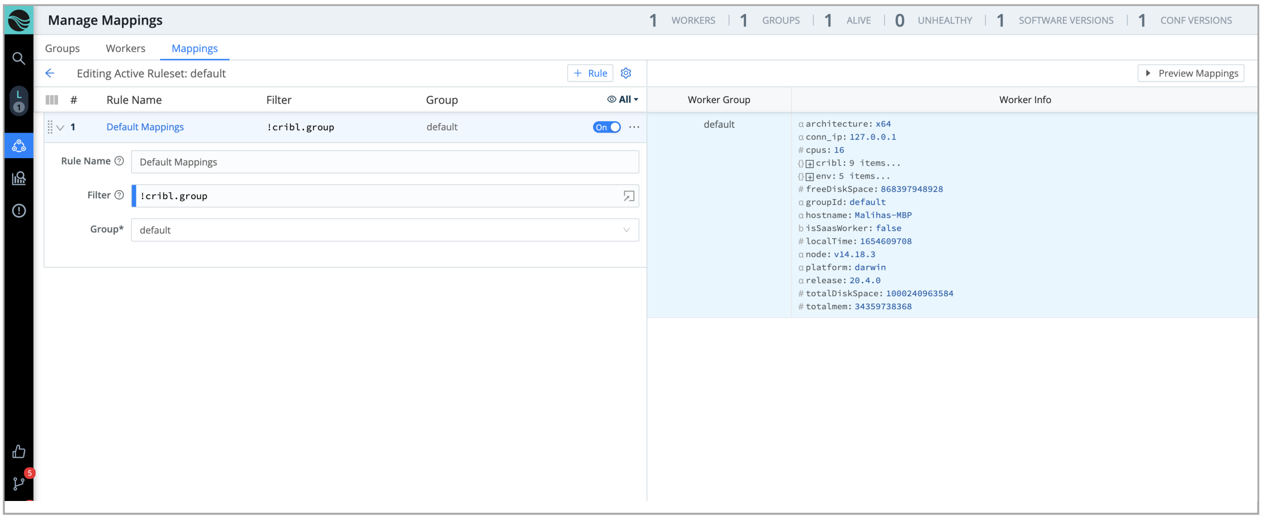
1. From the Leader's left nav, click **Manage**. On the **Manage Groups** page, click the **Mappings** tab.

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.)


Manage Mappings
1 WORKERS | 1 GROUPS | 1 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS
Groups
Workers
Mappings
Editing Active Ruleset: default
+ Rule
Preview Mappings
All
On

#	Rule Name	Filter	Group
1	Default Mappings	`!cribl.group`	default

Rule Name: Default Mappings

Filter: `!cribl.group`

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

Default Mapping Rule

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

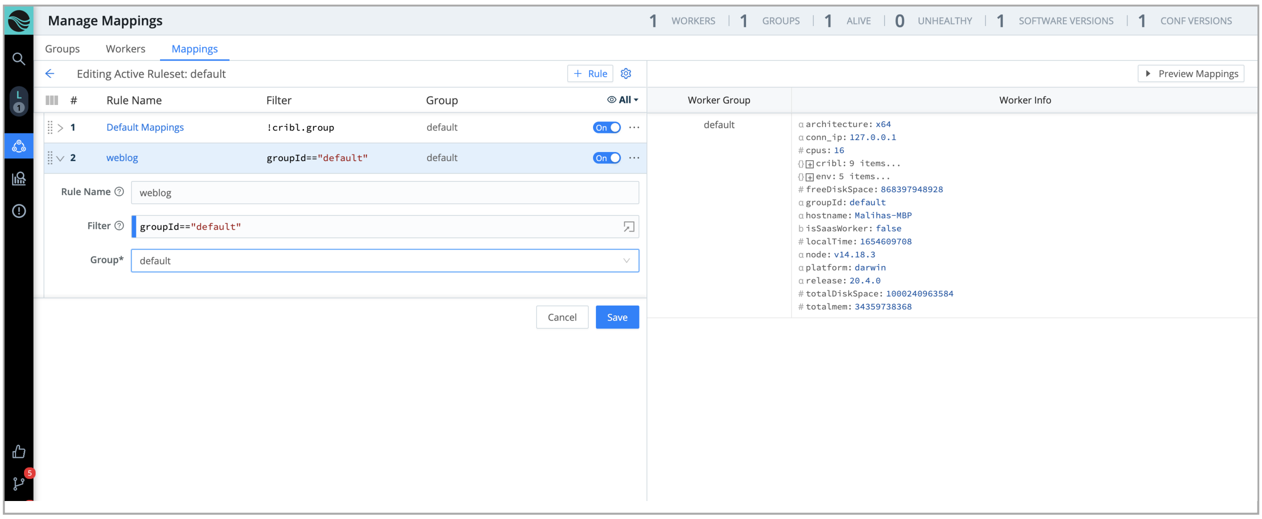
- **Filter**: `!cribl.group` – this value expression specifies "Not already assigned to a Worker Group."
- **Group**: `default` – this value specifies "Assign to the `default` Group."

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 **+ 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.


Manage Mappings
1 WORKERS | 1 GROUPS | 1 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS
Groups
Workers
Mappings
Editing Active Ruleset: default
+ Rule
Preview Mappings
All
Cancel
Save

#	Rule Name	Filter	Group	
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 left nav, select **Changes** > **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.

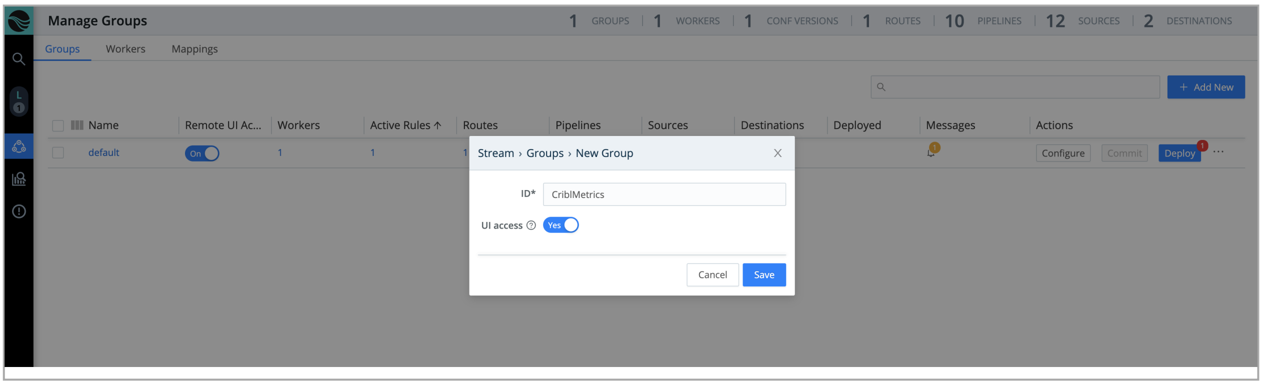
Add Another Worker Group

1. From the Leader's left nav, click **Manage**.

2. On the resulting **Manage Groups** page, click **+ Add New**.

3. Name the new Group `CriblMetrics` to match its purpose.

4. Set the slider for **UI access?** to `Yes`.


Manage Groups
1 GROUPS | 1 WORKERS | 1 CONF VERSIONS | 1 ROUTES | 10 PIPELINES | 12 SOURCES | 2 DESTINATIONS
Groups
Workers
Mappings
+ Add New
Configure
Commit
Deploy
Stream › Groups › New Group
Cancel
Save

Name	Remote UI Ac...	Workers	Active Rules ↑	Routes	Pipelines	Sources	Destinations	Deployed	Messages	Actions
default	On	1	1	1						Configure Commit Deploy

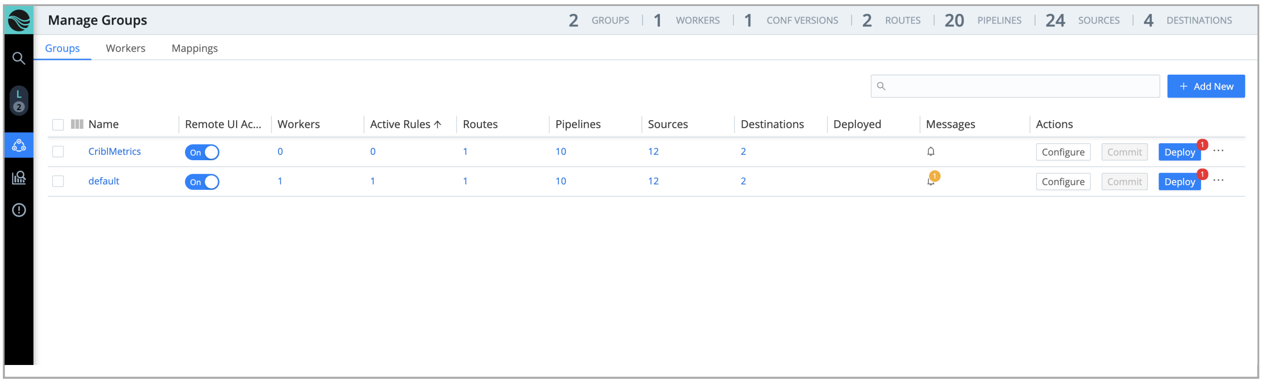
ID* CriblMetrics

UI access 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.


Manage Groups
2 GROUPS | 1 WORKERS | 1 CONF VERSIONS | 2 ROUTES | 20 PIPELINES | 24 SOURCES | 4 DESTINATIONS
Groups
Workers
Mappings
+ Add New

Name	Remote UI Ac...	Workers	Active Rules ↑	Routes	Pipelines	Sources	Destinations	Deployed	Messages	Actions
CriblMetrics	On	0	0	1	10	12	2			Configure Commit Deploy
default	On	1	1	1	10	12	2		1	Configure Commit Deploy

Second Groups saved, ready to Deploy

6. From the left nav, select **Changes** > **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.

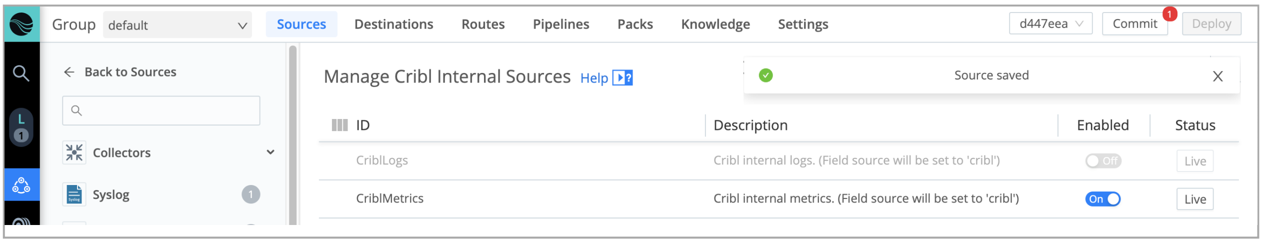
1. From the Leader's left nav, click **Manage** > `CriblMetrics`.

2. From the resulting top menu, click **Data** > **Sources**,.

3. From the **Data Sources** page's tiles or left menu, select **Cribl Internal**.

4. On the **Manage Cribl Internal Sources** page, slide the **CriblMetrics** slider to **On**.

5. Confirm that you want to enable CriblMetrics.



Manage Cribl Internal Sources

Source saved

ID	Description	Enabled	Status
CriblLogs	Cribl internal logs. (Field source will be set to 'cribl')	Off	Live
CriblMetrics	Cribl internal metrics. (Field source will be set to 'cribl')	On	Live

CriblMetrics Source enabled

6. From the left nav, select **Changes** > **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 left nav, click **Manage**.

2. From the **Manage Groups** page, click the **Mappings** tab.

3. Click the `default` Mapping Ruleset to open it.

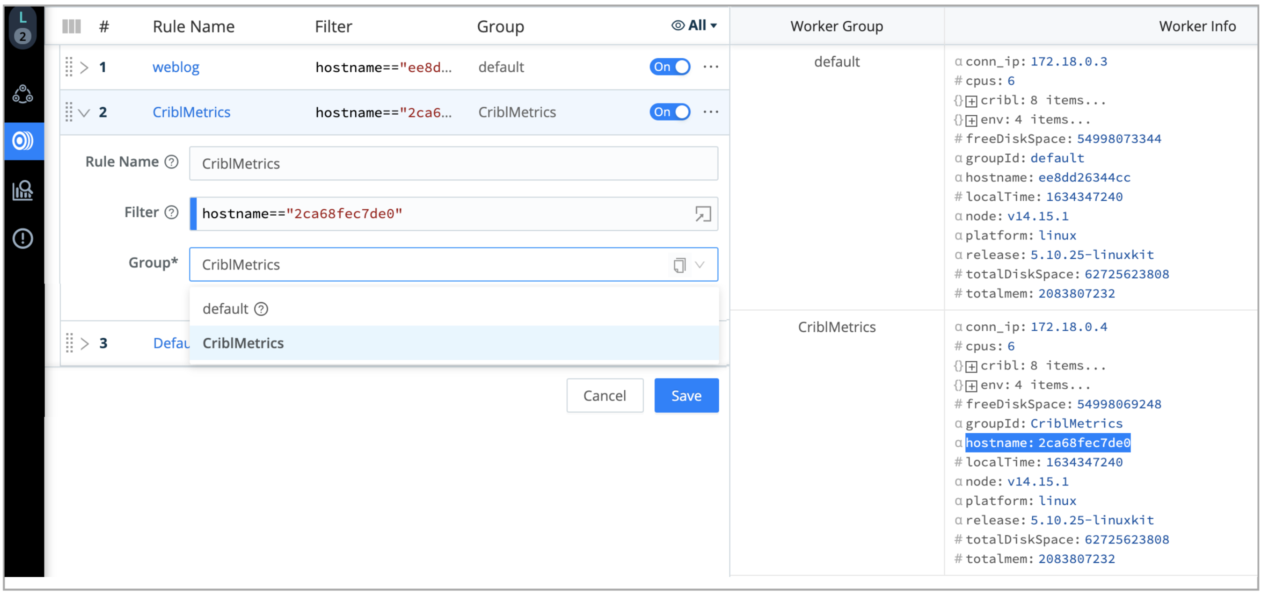
4. Click **+ Rule** at the upper right. Then configure the new Rule as shown below:

- **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.



#	Rule Name	Filter	Group	All
1	weblog	hostname=="ee8d...	default	On
2	CriblMetrics	hostname=="2ca6...	CriblMetrics	On
3	Defau			

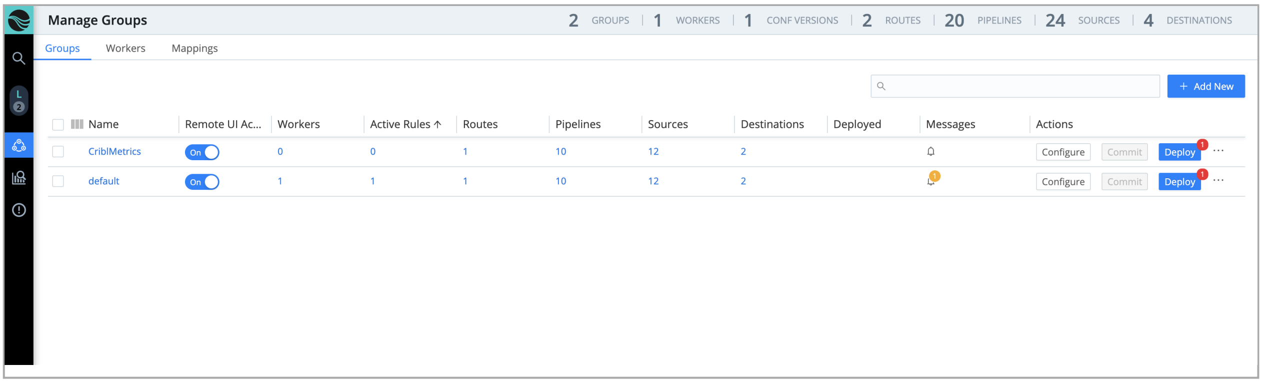
Rule Name	CriblMetrics
Filter	hostname=="2ca68fec7de0"
Group*	CriblMetrics

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 platform: linux release: 5.10.25-linuxkit totalDiskSpace: 62725623808 totalmem: 2083807232

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

8. From the left nav, select **Changes** > **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 left nav's **Manage** tab:



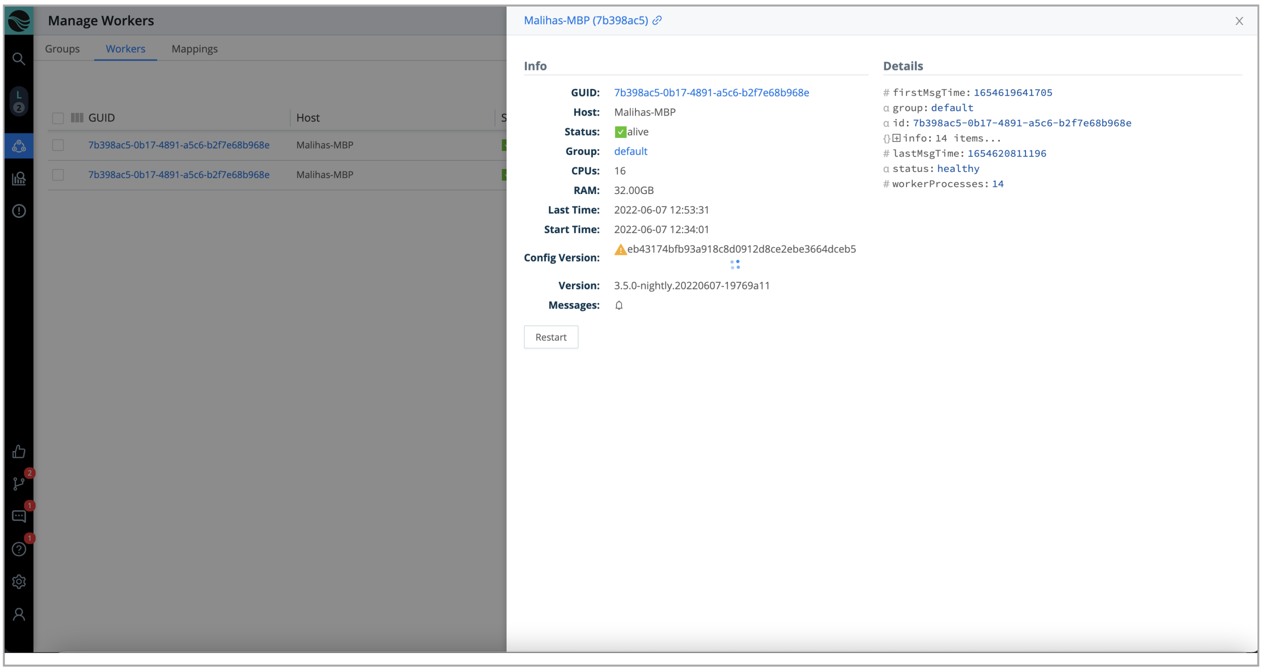
Manage Groups

2 GROUPS | 1 WORKERS | 1 CONF VERSIONS | 2 ROUTES | 20 PIPELINES | 24 SOURCES | 4 DESTINATIONS

Name	Remote UI Ac...	Workers	Active Rules ↑	Routes	Pipelines	Sources	Destinations	Deployed	Messages	Actions
CriblMetrics	On	0	0	1	10	12	2			Configure Commit Deploy
default	On	1	1	1	10	12	2		1	Configure Commit Deploy

Manage Groups page confirms Workers' mapping and assignment

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



GUID	Host
7b398ac5-0b17-4891-a5c6-b2f7e68b968e	Malihas-MBP
7b398ac5-0b17-4891-a5c6-b2f7e68b968e	Malihas-MBP

Malihas-MBP (7b398ac5)

Info

GUID: 7b398ac5-0b17-4891-a5c6-b2f7e68b968e
Host: Malihas-MBP
Status: ✓alive
Group: default
CPUs: 16
RAM: 32.00GB
Last Time: 2022-06-07 12:53:31
Start Time: 2022-06-07 12:34:01
Config Version: ⚠eb43174bfb93a918c8d0912d8ce2ebe3664dceb5
Version: 3.5.0-nightly.20220607-19769a11
Messages:

Details

```
# firstMsgTime: 1654619641705
α group: default
α id: 7b398ac5-0b17-4891-a5c6-b2f7e68b968e
{}⊞info: 14 items...
# lastMsgTime: 1654620811196
α 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.

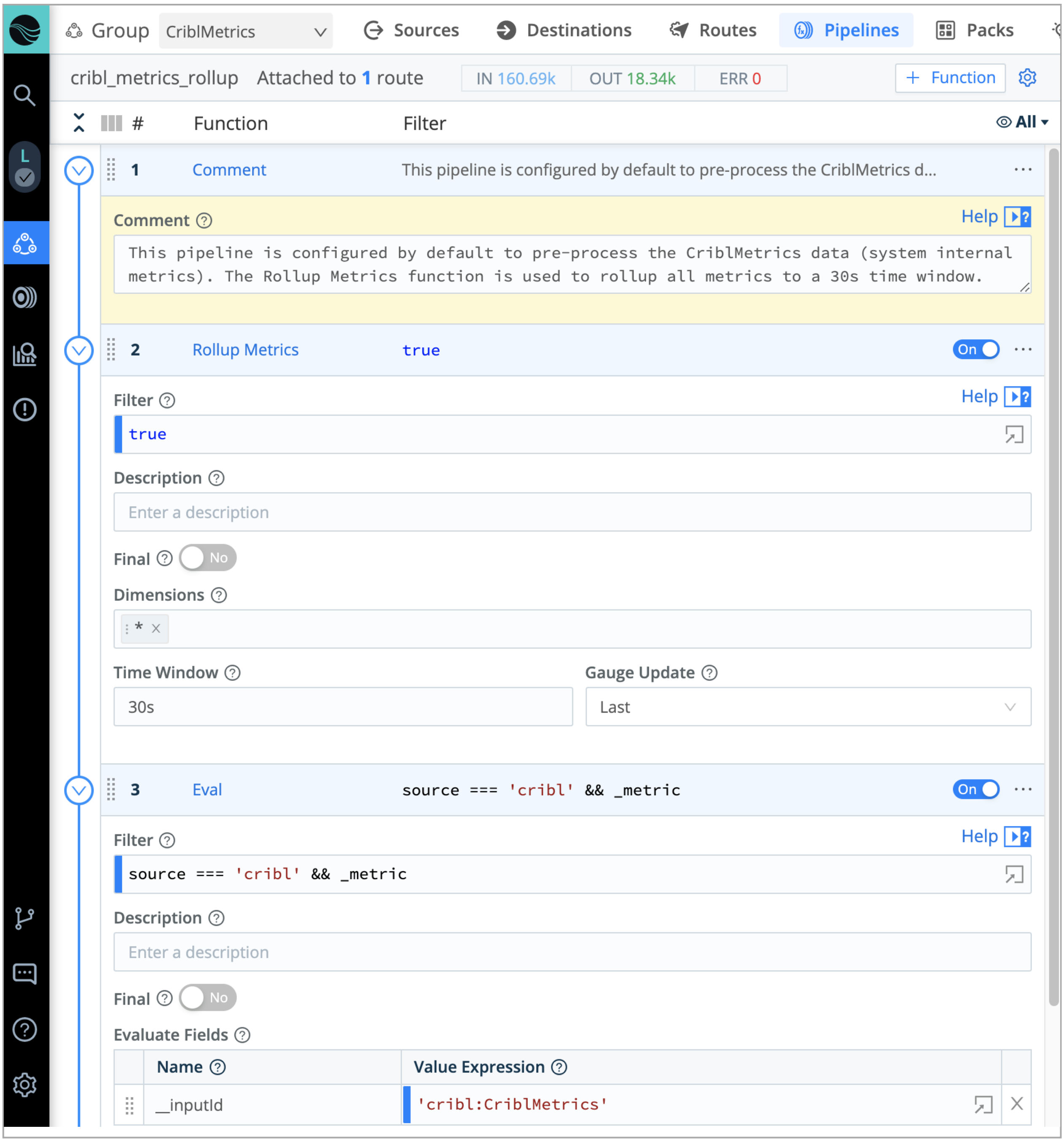
Examine the Metrics Pipeline

1. From the Leader's left nav, click **Groups** > `CriblMetrics`.

2. From the resulting top menu, select **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 Attached to 1 route

IN 160.69k OUT 18.34k ERR 0

#	Function	Filter
1	Comment	This pipeline is configured by default to pre-process the CriblMetrics d...
2	Rollup Metrics	true
3	Eval	source === 'cribl' && _metric

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

Filter: `true`

Description: Enter a description

Final: No

Dimensions: *

Time Window: 30s

Gauge Update: Last

3 Eval

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

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
__inputId	'cribl:CriblMetrics'

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 **Routes** page.

2. Click `+ 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`.

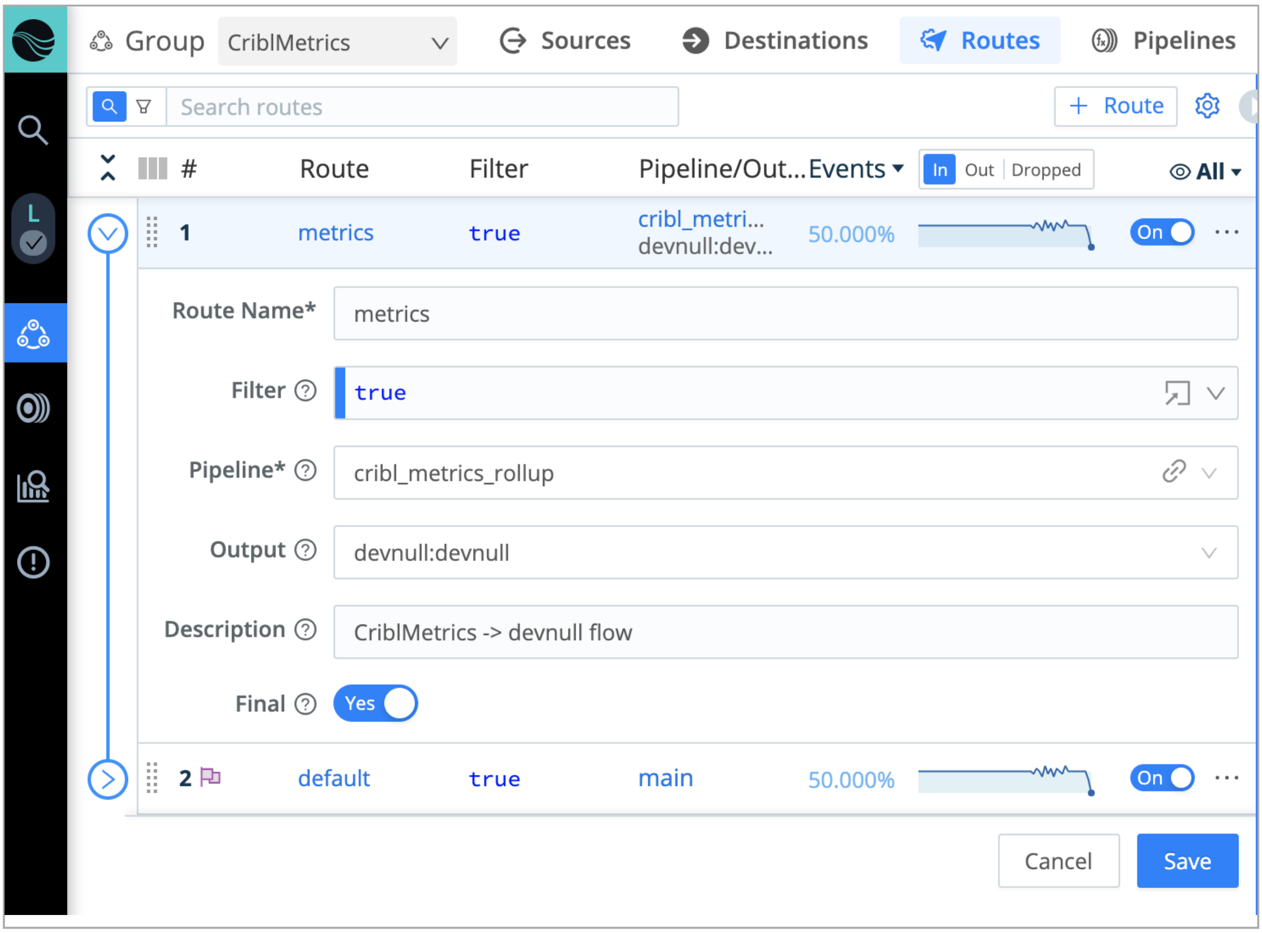
   This is Cribl Stream's preconfigured Destination that simulates a downstream service while simply dropping events.

7. You can leave the **Description** empty, and leave **Final** set to **Yes**.

8. Grab the new Route by its 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.



#	Route	Filter	Pipeline/Out...	Events	
1	metrics	true	cribl_metri... devnull:dev...	50.000%	On
2	default	true	main	50.000%	On

Route Name*: metrics

Filter: true

Pipeline*: cribl_metrics_rollup

Output: devnull:devnull

Description: CriblMetrics -> devnull flow

Final: Yes

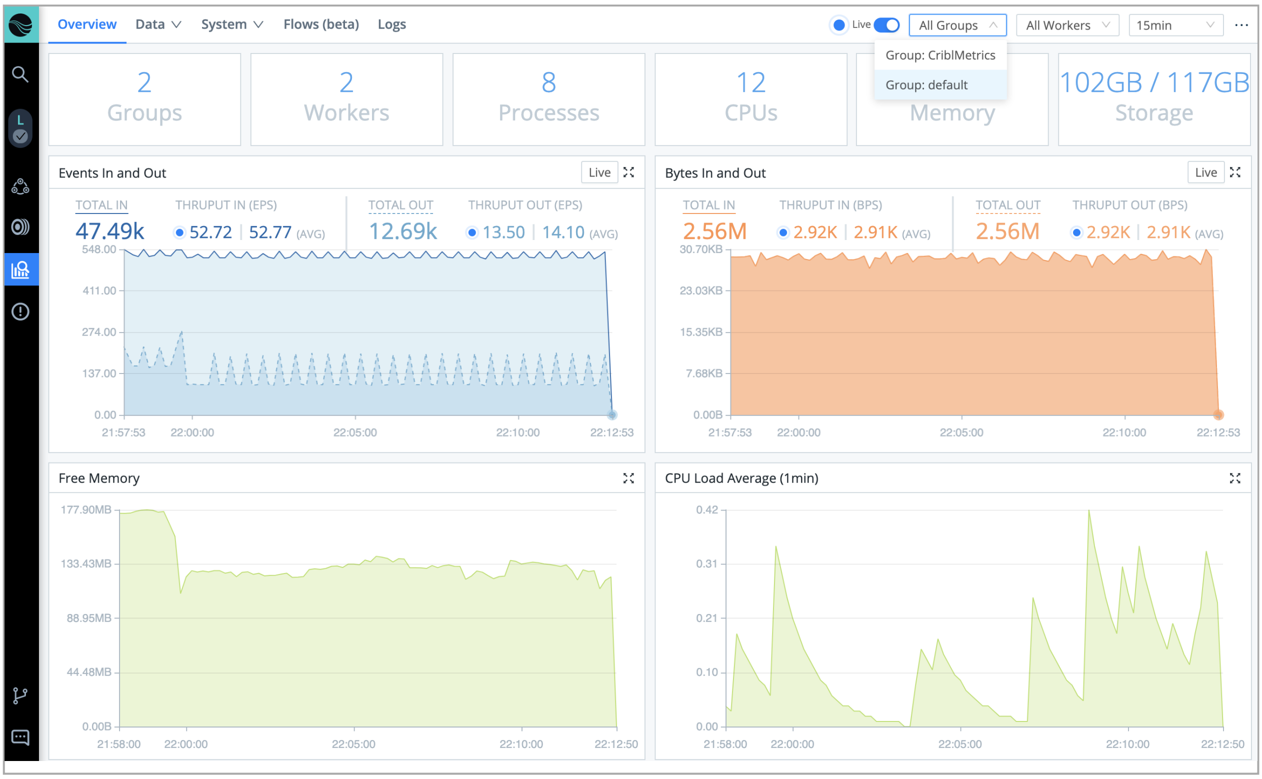
Cribl metrics Route, Pipeline, and Destination wired up

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 left nav's **Monitoring** link. 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:

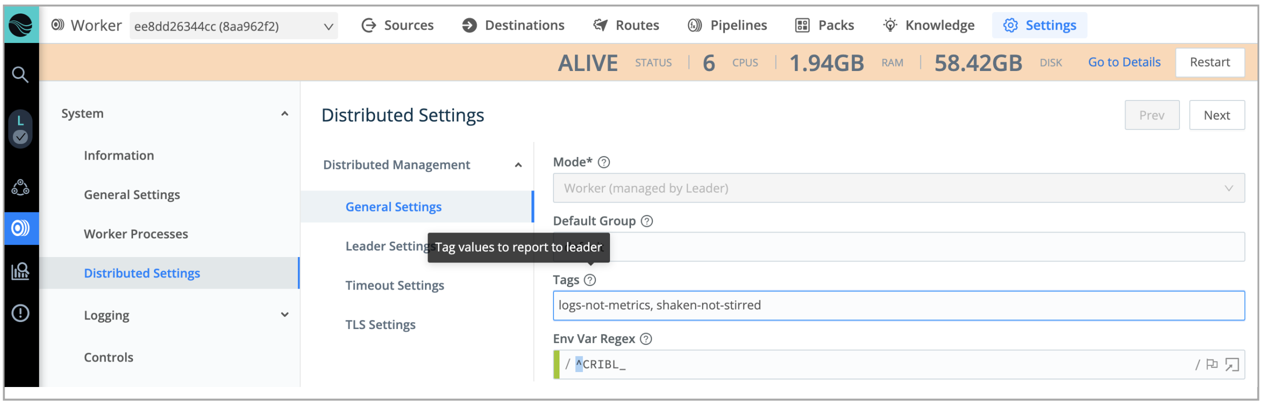
- A distributed deployment enables Cribl Stream to scale out to higher data volumes. This load-balancing occurs even with a single Worker Group, in which all Workers share the same configuration.

- 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*: 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 > **Sources** > **Datagens**, and switch off the slider beside `weblog.log`.
- If you configured a second Worker Group: Navigate to Cribl Stream's `CriblMetrics` Group > **Sources** > **Cribl Internal**, and switch off the slider 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.

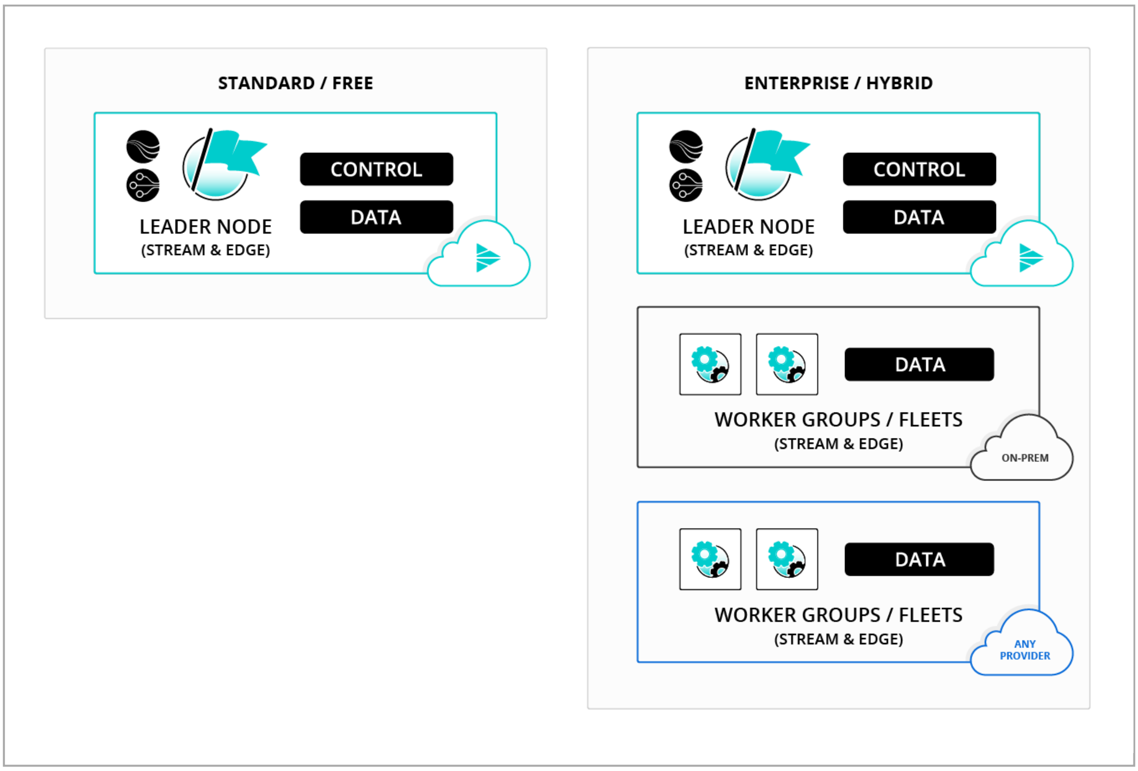
- Distributed Deployment: All the details behind the deployment approach you just mastered in this tutorial.
- Cribl Stream Sandboxes: Work through general and specific scenarios in a free, hosted environment, with terminal access and real data inputs and outputs.
- Use Cases documentation: Bring your 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. Launch Guide

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

By upgrading to an Enterprise plan, you can expand this to a **hybrid deployment** of any desired complexity. Here, the Leader (the control plane) resides in Cribl.Cloud, while the Workers that process the data (the data plane) can be variously in Cribl.Cloud, in cloud instances that you manage, and/or in your data centers.



Standard/free versus Enterprise/hybrid deployment

> For an overview of additional features available on Enterprise plans, see Pricing.

Why Use Cloud Deployment? (Advantages)

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

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

Getting Started

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

Your Organization will display a dedicated Portal, a network and access boundary that isolates your Cribl resources from all other users. Each Cribl.Cloud account provisions a separate AWS account. Your Cribl Stream instance is deployed inside a virtual private cloud (VPC) in this account.

The Portal will initially contain a free Cribl Stream instance. 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.

About Cribl Stream (and This Document)

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

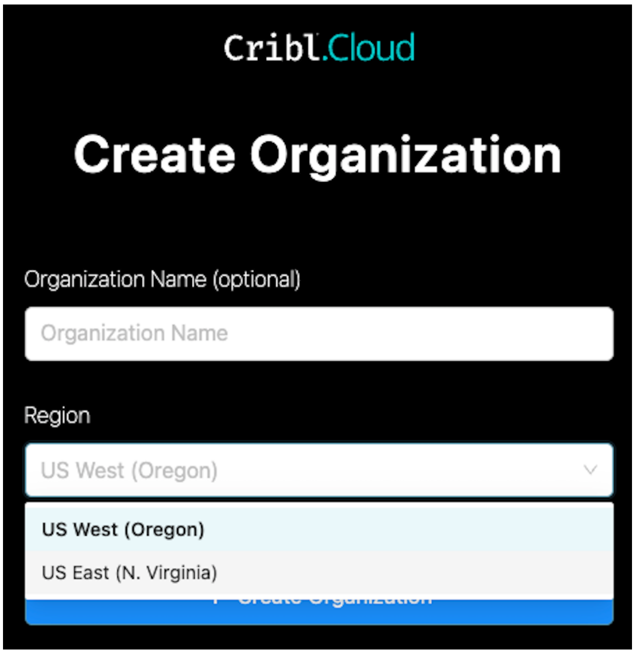
> Cribl.Cloud always runs in distributed mode – see Simplified Distributed Architecture below for details.

Registering a Cribl.Cloud Portal

Ready to take the red pill? The next few sections explain how to register and manage a Cribl.Cloud instance.

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

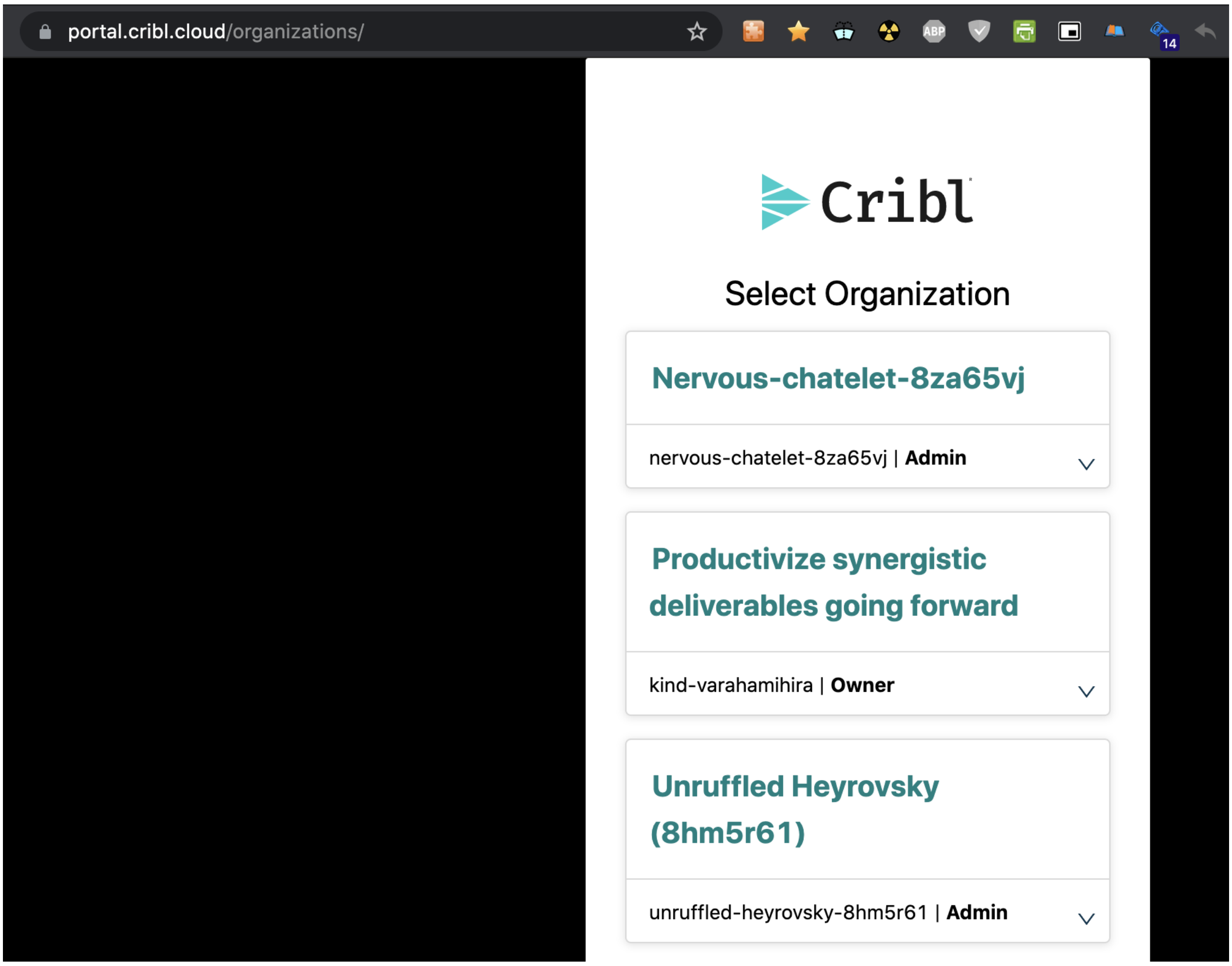
1. Start at: https://cribl.cloud/signup/
2. Select the **New User? Free signup** option, and register with your work email address.
3. Use the verification code from Cribl's email to confirm your registration.
4. On the **Create Organization** page, optionally enter an **Organization Name** (a friendly alias for the randomly generated ID that Cribl will assign to your Organization).
5. Select an AWS Region to host your Cribl.Cloud Leader and Cribl-managed Workers. Cribl currently supports either the `US West (Oregon)` or `US East (Virginia)` Region.
6. Bookmark your Cribl.Cloud portal page, for all that follows.



Create Organization page – selecting a host Region

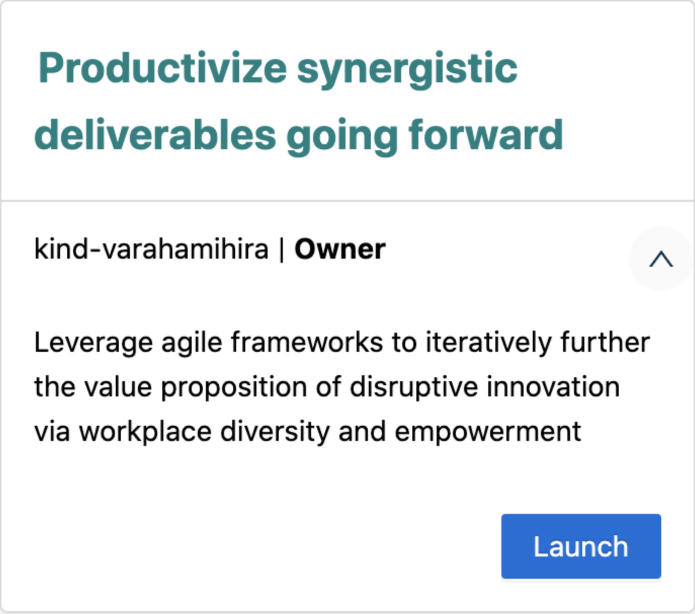
Select Organization Page

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


portal.cribl.cloud/organizations/
Cribl
Select Organization
Nervous-chatelet-8za65vj
nervous-chatelet-8za65vj | **Admin**
Productivize synergistic deliverables going forward
kind-varahamihira | **Owner**
Unruffled Heyrovsky (8hm5r61)
unruffled-heyrovsky-8hm5r61 | **Admin**

Select Organization interstitial page

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


Productivize synergistic deliverables going forward
kind-varahamihira | **Owner**
Launch

Leverage agile frameworks to iteratively further the value proposition of disruptive innovation via workplace diversity and empowerment

Organization tile's details and controls

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

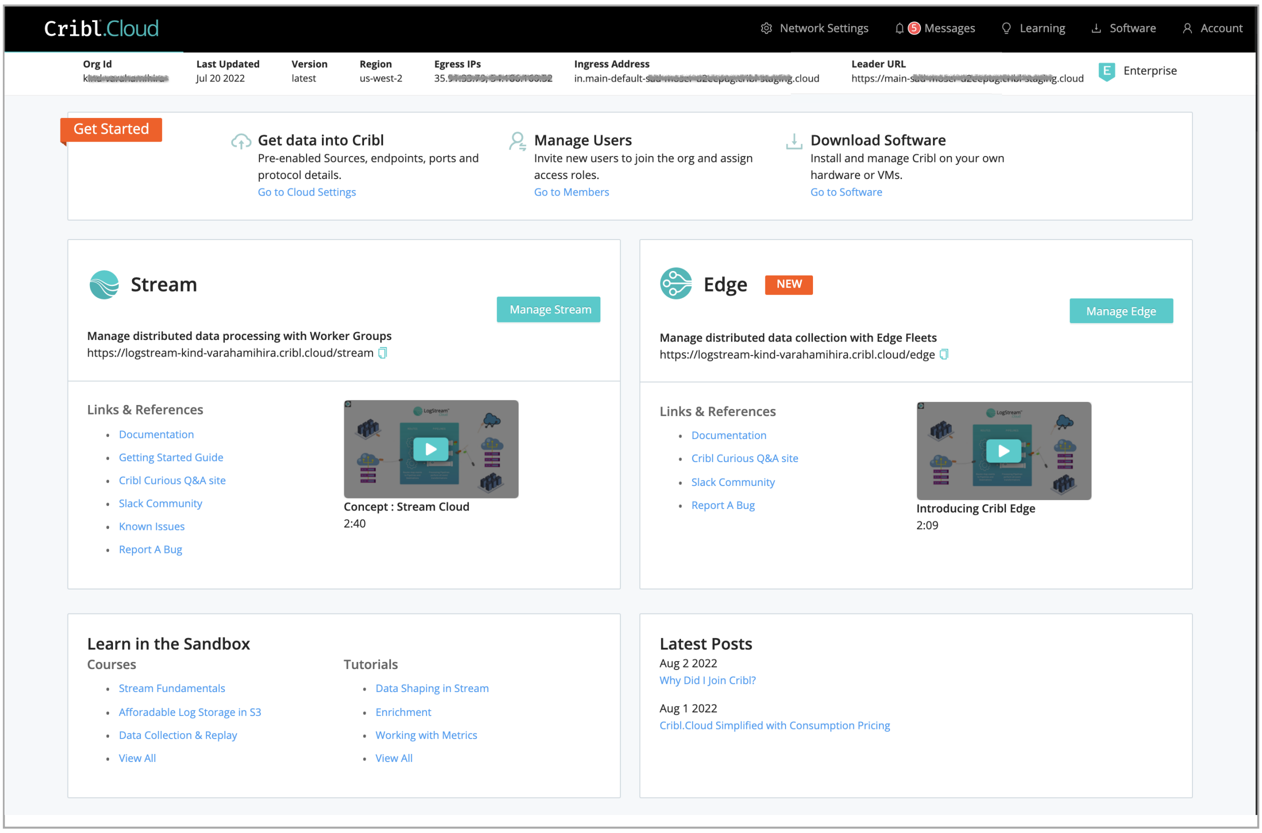
Exploring the Cribl.Cloud Portal

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

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

Portal Page (Cribl.Cloud Logo)

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



Org Id	Last Updated	Version	Region	Egress IPs	Ingress Address	Leader URL	
k[redacted]	Jul 20 2022	latest	us-west-2	35.[redacted]	in.main-default-[redacted]g.cloud	https://main-[redacted]g.cloud	Enterprise

Get Started

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 Stream
Manage distributed data processing with Worker Groups
https://logstream-kind-varahamihira.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 NEW
Manage Edge
Manage distributed data collection with Edge Fleets
https://logstream-kind-varahamihira.cribl.cloud/edge

Links & References
- Documentation
- Cribl Curious Q&A site
- Slack Community
- Report A Bug

Introducing Cribl Edge
2:09

Learn in the Sandbox

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

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

Latest Posts

Aug 2 2022
Why Did I Join Cribl?

Aug 1 2022
Cribl.Cloud Simplified with Consumption Pricing

Cribl.Cloud portal

However, the surrounding page offers lots more useful information:

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

Overview Strip

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

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

Last Updated: Date on which Cribl last pushed an infrastructure change (notably including changes to the above **Egress Address**).

Version: Your deployed Cribl Stream version.

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

Egress IPs: Your Cribl.Cloud instance's current public IP address. This address is dynamic; Cribl will occasionally update it when we need to rescale core infrastructure.

Ingress Address: Your Cribl.Cloud instance's global domain for inbound data (before specifying ports per data type).

Network Settings Page

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

Data Sources Tab

The **Data Sources** tab lists ports, protocols, and data ingestion inputs that are open and available to use. Return to this tab to copy Ingest Addresses (endpoints) as needed. For details, see Available Ports and TLS Configurations.

ACL Tab

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.

Trust Tab

The **Trust** tab provides a **Worker ARN** (Amazon Resource Name) that you can copy and paste to attach a Trust Relationship to an AWS account's IAM role. Doing so 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.

Cribl Stream UI Access

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

Messages Drawer

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

Learning Page

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

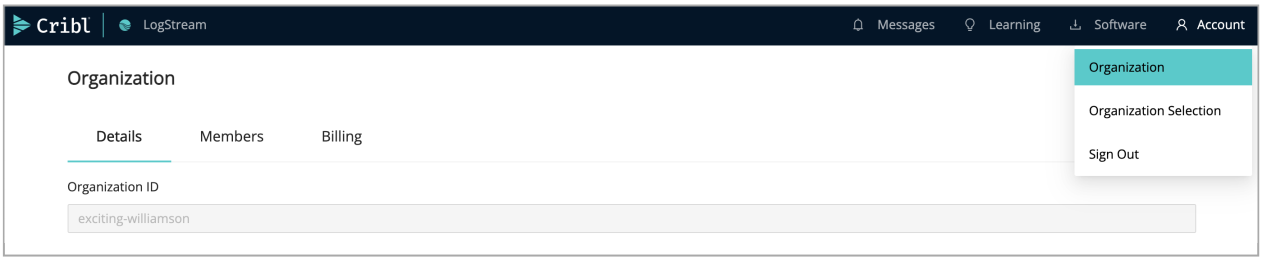
- Sandboxes (free, interactive tutorials on fully hosted integrations).
- Documentation.
- Product and plans overview (pricing comparison).
- Cribl events (including future and archived Webinars).
- Concept/demo videos.

Software Page

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

Account Menu

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



Organization ID

exciting-williamson

Account tab

Organization Page

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

Details Tab

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

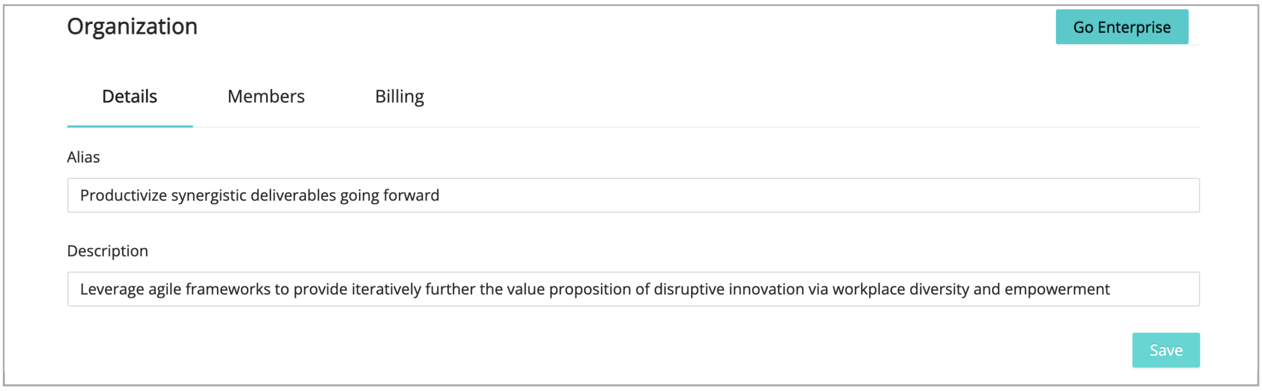
Alias: Optionally, enter a "friendly" name for your Organization. Upon signing in, members will see this alias above the Organization ID on the Select Organization page.

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

License: This drop-down displays one or multiple licenses that you've currently applied to 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 Cloud.

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



Alias

Productivize synergistic deliverables going forward

Description

Leverage agile frameworks to provide iteratively further the value proposition of disruptive innovation via workplace diversity and empowerment

Organization Details tab

Members Tab

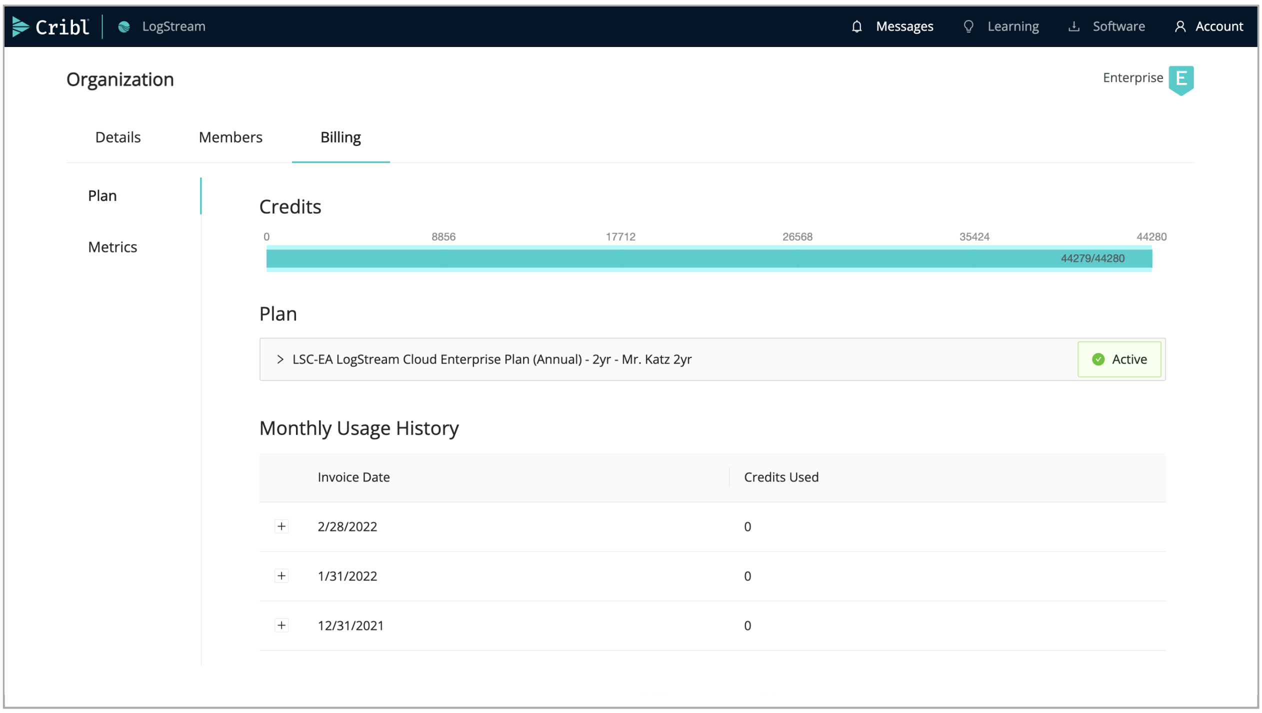
The **Organization** > **Members** upper tab provides access to inviting and managing other users.

Billing Tab

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

Plan Tab

The **Plan** left tab displays a mercury bar of available **Credits** on your account, an expandable **Plan** details accordion, and expandable **Monthly Usage History** rows offering details about your data throughput volume in current and prior months.



> LSC-EA LogStream Cloud Enterprise Plan (Annual) - 2yr - Mr. Katz 2yr — Active

Monthly Usage History

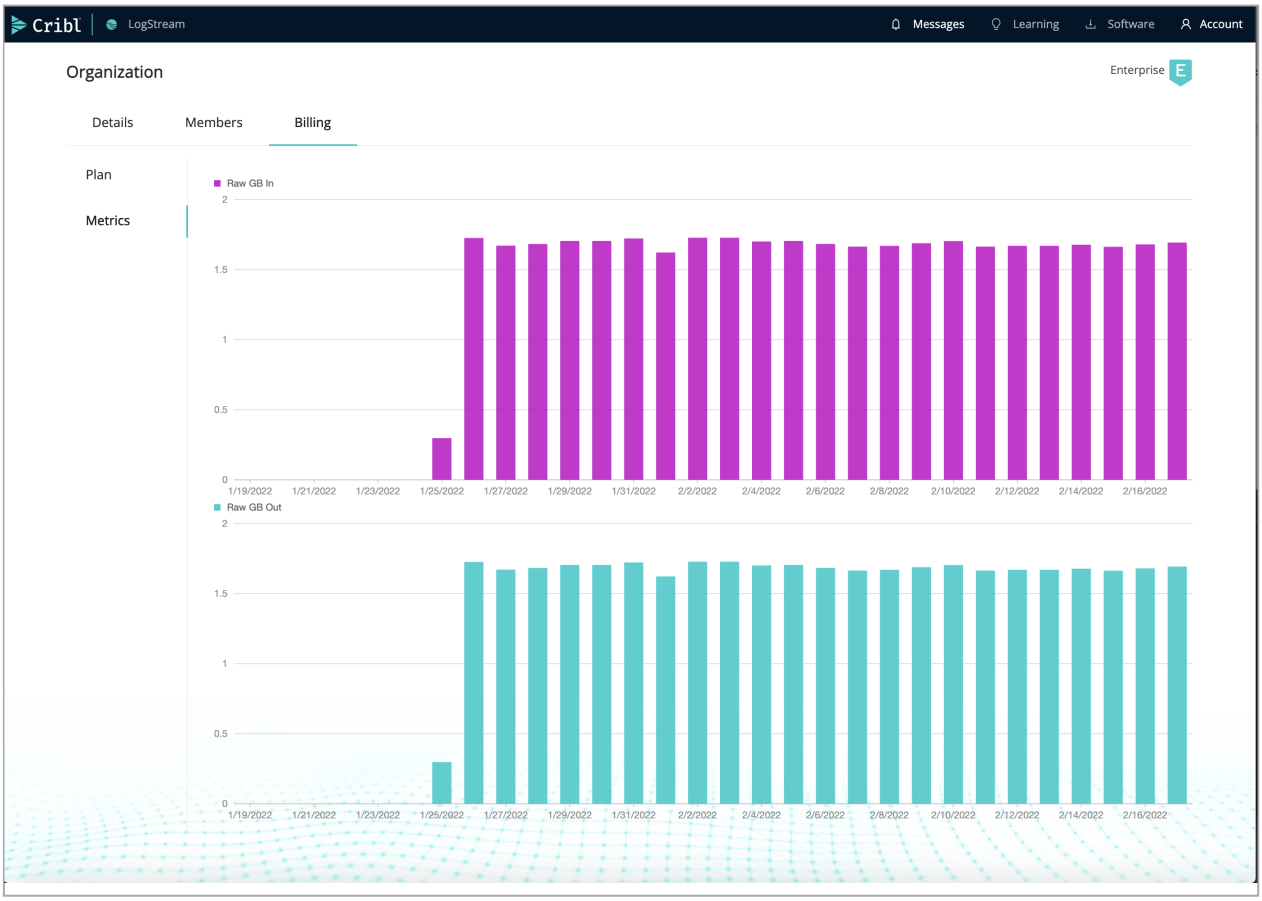
	Invoice Date	Credits Used
+	2/28/2022	0
+	1/31/2022	0
+	12/31/2021	0

Billing > Plan tab

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

Metrics Tab

The **Metrics** left tab provides bar graphs showing **Raw GB In** and **Raw GB Out** for each day over the last month.



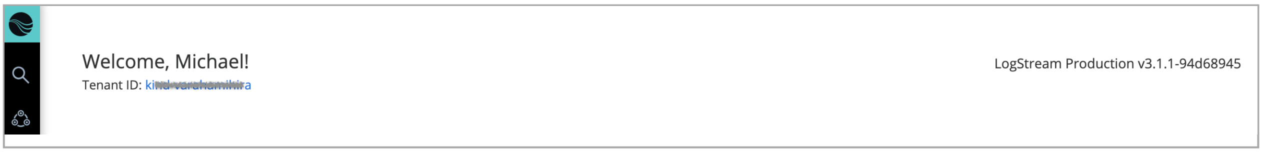
Billing > Metrics tab

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 either **Manage Stream** or **Manage Edge**.
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.



Welcome, Michael!

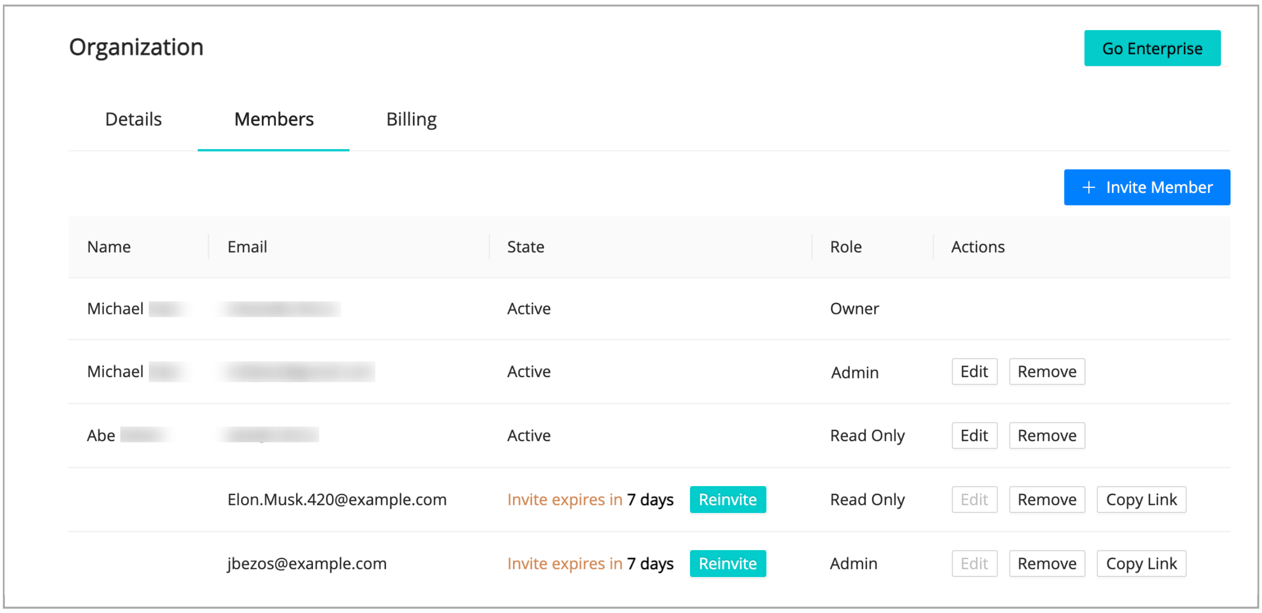
Tenant ID: ki[redacted]a

LogStream Production v3.1.1-94d68945

Cribl.Cloud link takes you back to the portal

Inviting and Managing Other Users

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



Organization

Go Enterprise

Details | Members | Billing

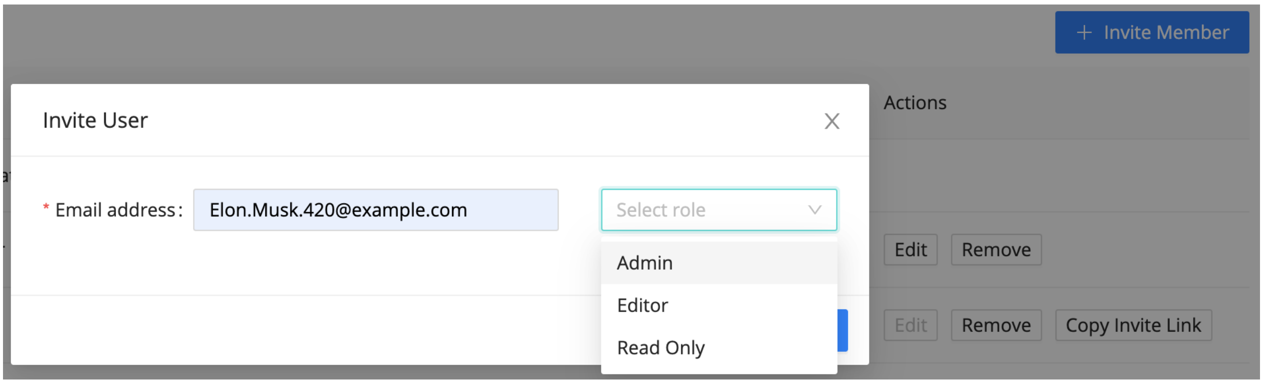
+ Invite Member

Name	Email	State	Role	Actions
Michael [redacted]	[redacted]	Active	Owner	
Michael [redacted]	[redacted]	Active	Admin	Edit Remove
Abe [redacted]	[redacted]	Active	Read Only	Edit Remove
	Elon.Musk.420@example.com	Invite expires in 7 days Reinvite	Read Only	Edit Remove Copy Link
	jbezos@example.com	Invite expires in 7 days Reinvite	Admin	Edit Remove Copy 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 a **Role** (explained just below), and then click **Invite** to send the invitation.



Invite User

* Email address: Elon.Musk.420@example.com

Invite User modal

Member Roles

Each Role that you can assign to members confers a default Role within the Organization's Cribl.Cloud instance. Here are the Roles, their corresponding permissions, and who can assign each:

MEMBER ROLE	CRIBL STREAM ROLE	OPTIONS/RESTRICTIONS
`Admin`	`admin`	Any Organization owner can assign
`Editor`	`editor_all`	Assignable only with Enterprise plan
`Read-Only`	`reader_all`	Assignable only with Enterprise plan
`Owner`	`admin`	Can't be assigned, but can manage Organization details

Note that:

- Owners of non-Enterprise Cribl.Cloud Organizations can assign only the `Admin` Role in the **Invite User** modal shown above.
- Expanded role-based access control – i.e., the ability to manage the `Editor` and `Read-Only` Roles shown above – is available only with an Enterprise plan. (For all available Enterprise features, see Pricing.)
- The one Member Role that you cannot assign or transfer is your own Owner Role. A user can acquire this superuser Role only by signing up as the owner of their own Cribl.Cloud Organization.
- Only an Organization's Owner can manage the Organization's details.
- You assign Roles per individual user, when you invite them to your Organzation. Cribl.Cloud does not currently support globally predefining or assigning group Roles, as with on-premises Cribl Stream. However, Admins can change users' Roles after those users join their Organization.

> ### Cribl.Cloud Roles Rule Cribl Stream Access
>
> When you assign a Cribl.Cloud Member Role, it is mapped to a Cribl Stream Role as described above. However, these users will **not** be visible as local users within the UI of Cribl Stream Cloud instances managed by Cribl.
>
> Also, within these instances' UI: modifying Roles **not** mapped above will have no effect; and adding local users wil have no effect.

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.

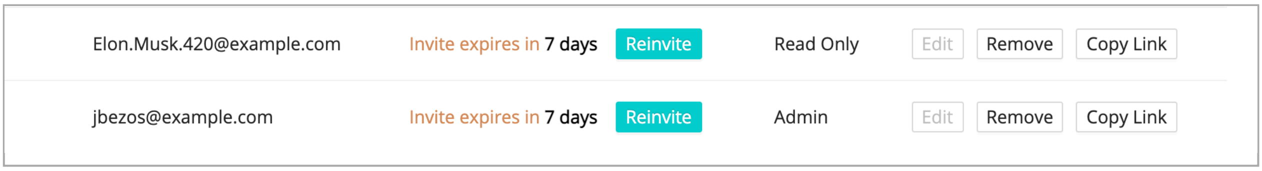
After signing in, they'll have access to your Organization and Cribl Stream instance at the Role level you've specified.

Managing Invites

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

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

After 7 days, if an invite has been neither accepted nor revoked, it expires. In this case, it is removed from the **Members** tab.



Elon.Musk.420@example.com	Invite expires in 7 days	Reinvite	Read Only	Edit	Remove	Copy Link
jbezos@example.com	Invite expires in 7 days	Reinvite	Admin	Edit	Remove	Copy Link

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 Role. (The **Edit** option is displayed only if you have an Enterprise plan.)

- **Remove**: Remove this member from your Organization. After clicking this button, you'll see a confirmation dialog. (Proceeding will not affect this user's access to any other Cribl.Cloud Organizations they might own or be members of.)

Cloud Pricing

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

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

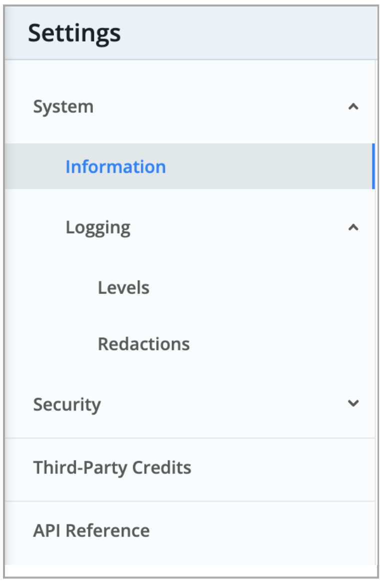
Differences from Self-Hosted Cribl Stream

A Cribl.Cloud deployment can differ from an on-premises/customer-managed Cribl Stream deployment in the following ways. 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 left nav and Settings page.



Cribl.Cloud's 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. With a Free or Standard plan, there is a single Worker button and Worker.

Compared to self-hosted Cribl Stream, the **Settings** > **Worker Processes** and **Settings** > **Distributed Settings** links are omitted, and the left nav contains no **Worker Groups** or **Mappings** links.

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 Workers and Worker Groups.

Git Preconfigured

Without an Enterprise plan, the **Settings** > **System** > **Git Settings** section is omitted. A local `git` client is preconfigured in your Cribl.Cloud portal. On Cribl.Cloud's left nav, use the **Changes** link to commit/push changes to `git`. Select **Deploy** at the UI's top right 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

Without an Enterprise plan, the **Settings** > **Access Management** section is omitted. All users of a given Cribl.Cloud instance share a single `admin` login.

Without an Enterprise plan, the **Settings** > **Security** > **KMS** section is omitted. The free version does not support KMS secrets stores.

If you add an Enterprise Plan, Cloud and hybrid Leaders support Local and Google SSO authentication. However, they do not currently support other OpenID Connect authentication schemes, nor LDAP, nor the SAML protocol.

Role-based access control (RBAC) is simplified in Cribl.Cloud. For details, see Member Roles.

Transparent Licensing

The left nav's **Settings** > **Licensing** link is omitted. Your license is managed by your parent Cribl.Cloud portal, where you can check credits and usage history on the Billing tab.

Other Simplified Settings

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

- The Script Collector, and custom Functions, are available only on hybrid, customer-managed Workers. (These features are currently not available on Cribl-managed Workers.)
- The left nav's **Settings** > **Scripts** link is omitted from Cribl.Cloud, which currently does not support configuring or running shell scripts on hybrid or Cribl-managed Workers.
- The Filesystem Collector and Filesystem Destination are available only on hybrid Workers. (Cribl-managed Workers have no local filesystem to read from or write to.)
- Persistent Queues can be configured only on hybrid (not Cribl-managed) Workers' Sources and Destinations.
- File-based Destinations support staging directories only on hybrid (not Cribl-managed) Workers.
- The Tee Function is available only on hybrid (not Cribl-managed) Workers.

Support Options

The **Settings** > **Diagnostics** link is omitted. For help with any troubleshooting needs:

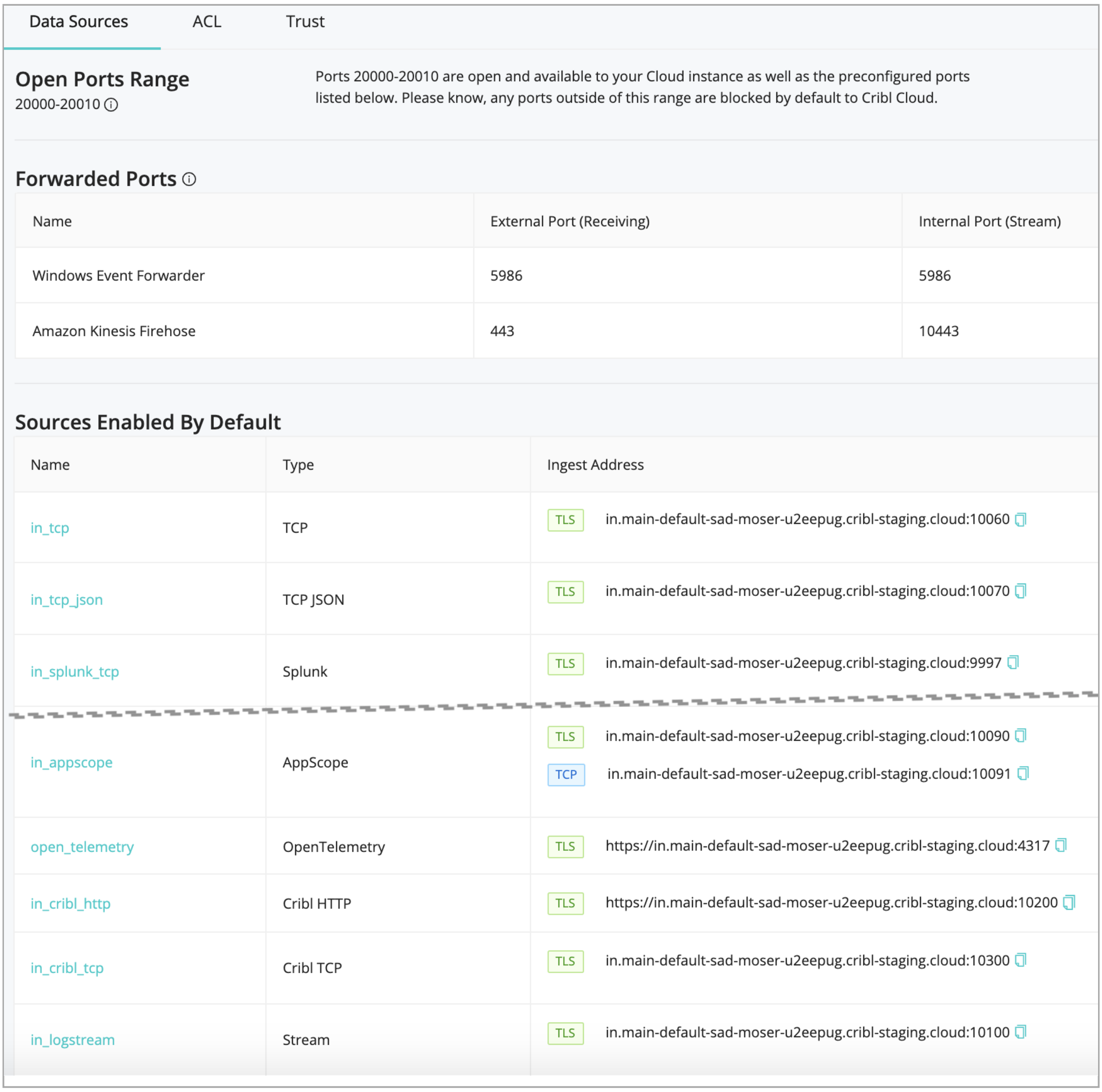
- Click the Intercom link at Cribl Stream's lower right.
- Join Cribl's Community Slack `#cribl-cloud` channel.

- If you have a paid account, contact Cribl Support.

Available Ports and TLS Configurations

To get data into Cribl.Cloud, your Cribl.Cloud portal provides several data sources and ports already enabled for you, plus 11 additional ports (`20000-20010`) that you can use to add and configure more Cribl Stream 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 the existing Sources listed here, Cribl recommends using the preconfigured endpoint and port to send data into Cribl Stream.



Open Ports Range
20000-20010

Ports 20000-20010 are open and available to your Cloud instance as well as the preconfigured ports listed below. Please know, any ports outside of this range are blocked by default to Cribl Cloud.

Forwarded Ports

Name	External Port (Receiving)	Internal Port (Stream)
Windows Event Forwarder	5986	5986
Amazon Kinesis Firehose	443	10443

Sources Enabled By Default

Name	Type	Ingest Address
in_tcp	TCP	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10060
in_tcp_json	TCP JSON	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10070
in_splunk_tcp	Splunk	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:9997
in_appscope	AppScope	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10090 TCP in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10091
open_telemetry	OpenTelemetry	TLS https://in.main-default-sad-moser-u2eepug.cribl-staging.cloud:4317
in_cribl_http	Cribl HTTP	TLS https://in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10200
in_cribl_tcp	Cribl TCP	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10300
in_logstream	Stream	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10100

Available ports and TLS certificates

TLS Details

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

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

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

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

> **Cribl HTTP and Cribl TCP Sources/Destinations**
>
> Use the Cribl HTTP Destination and Source, and/or the Cribl TCP Destination and Source, to relay data between Worker Nodes connected to the same Leader. This traffic does not count against your ingestion quota, so this routing 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 Cribl Internal Source is omitted from Cribl.Cloud instances, because Cribl manages these instances' uptime and diagnostics on your behalf. Also, the Exec Source, available in self-hosted v.3.3 and above, is unavailable in Cribl.Cloud instances.

> 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.

Enterprise Cloud

With an Enterprise plan, Cribl.Cloud offers the same options and flexibility as a customer-managed Cribl Stream distributed deployment with an Enterprise license – and more.

These options include configuring and managing multiple Worker Groups or Fleets,maintaining version control with remote repos, Notifications, Google SSO authentication, and Role-based access control to Cribl Stream resources.

> For other Enterprise features, see Pricing.

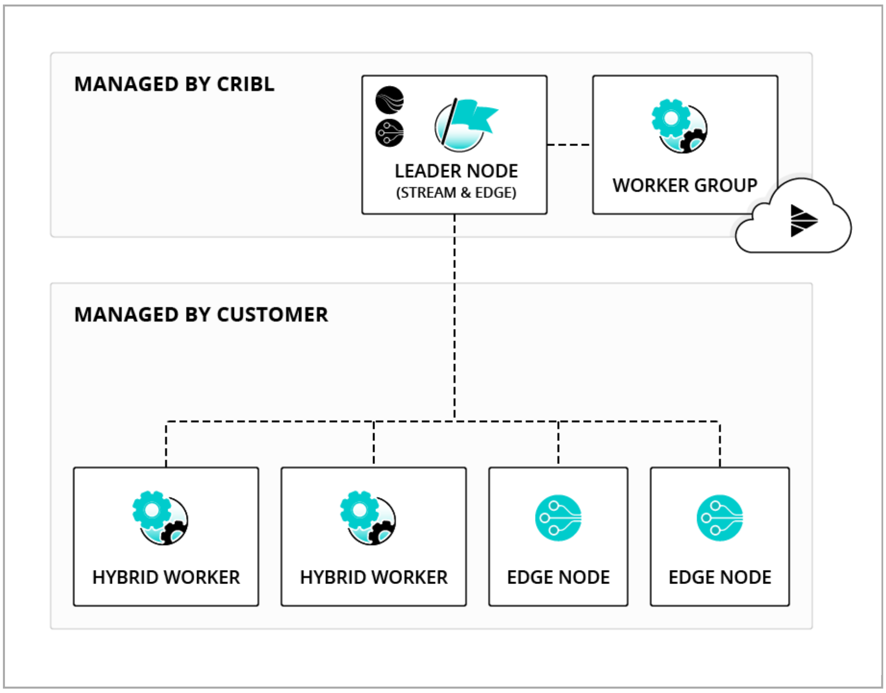
Cribl.Cloud Enterprise also adds:

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

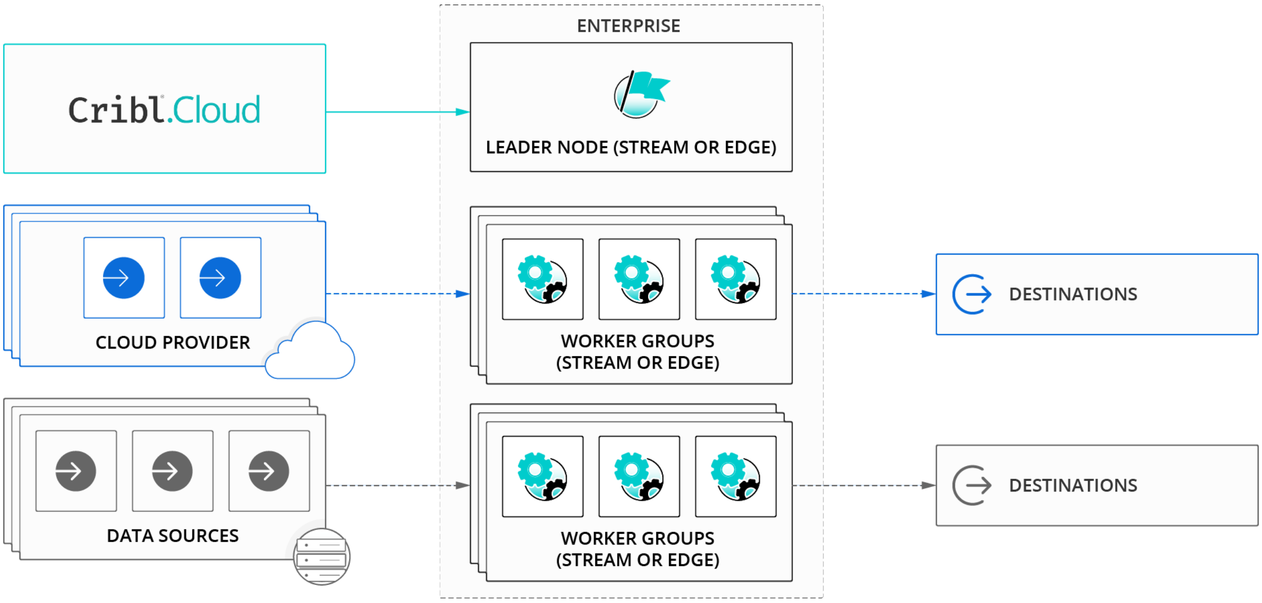
Hybrid Deployment

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

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



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



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

As the footprint of your operations grows or changes, this flexibility makes it easy to reconfigure Cribl 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, and reassign them to different Worker Groups, by easily auto-generating 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-premises, 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 Workers.
- On all Workers' hosts, port 4200 must be open for management by the Leader.
- On all Workers' hosts, firewalls must allow outbound communication on port 443 to the Cribl.Cloud Leader, and on port 443 to https://cdn.cribl.io.
- If this traffic must go through a proxy, see System Proxy Configuration for configuration details.

Note that you are responsible for data encryption and other security measures on Worker instances that you manage.

Adding (Bootstrapping) Workers

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

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

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

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

Hybrid Cribl HTTP/Cribl TCP Configuration

If you use the Cribl HTTP Destination and Source pair, or the Cribl TCP Destination and Source pair, to relay data between 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.

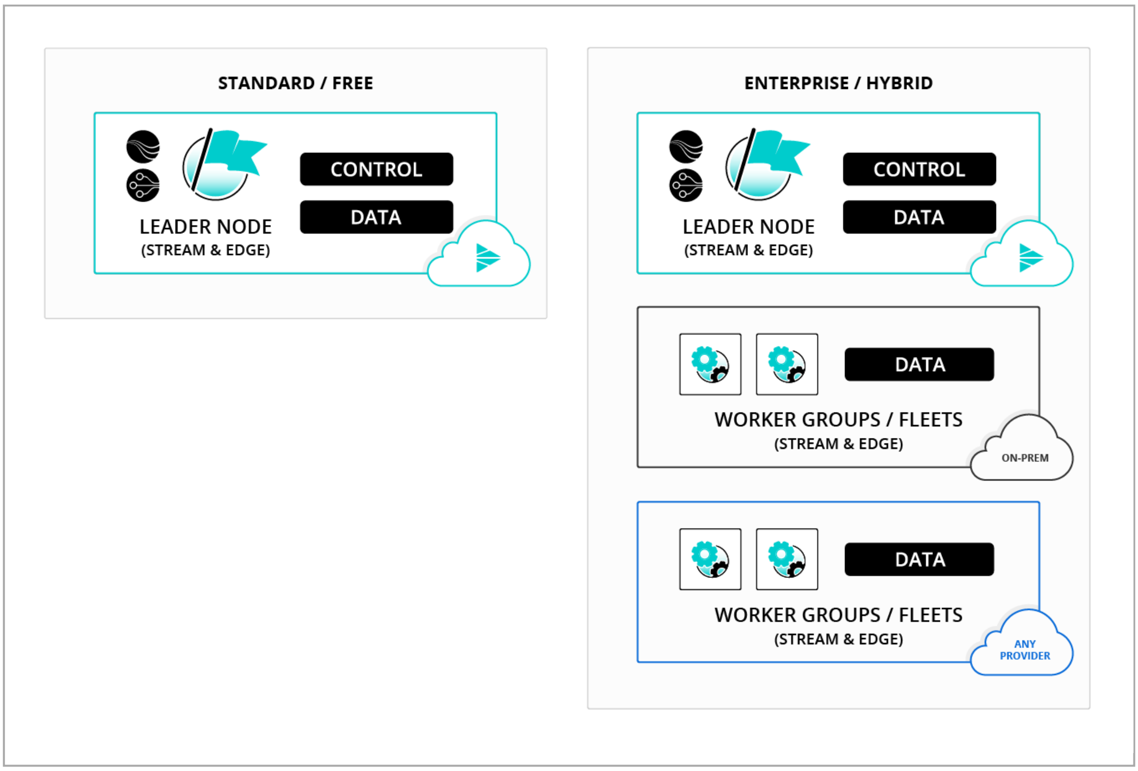
- Configure hybrid Workers by logging directly into their UI, then selecting global ⚙ **Settings** (lower left) > **Distributed Settings**. Make sure the **Mode** is set to **Managed Worker** or **Managed Edge** (which might require a restart).
- Then select the **Leader Settings** left tab, and ensure a consistent entry in the **Address** field.
- In Cloud hybrid deployments, the Leader's Address format is `main-<your-Org-ID>.cribl.cloud.` When configuring a hybrid Worker, use that format in the **Address** field.

;

2.1. Launch Guide

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

By upgrading to an Enterprise plan, you can expand this to a **hybrid deployment** of any desired complexity. Here, the Leader (the control plane) resides in Cribl.Cloud, while the Workers that process the data (the data plane) can be variously in Cribl.Cloud, in cloud instances that you manage, and/or in your data centers.



Standard/free versus Enterprise/hybrid deployment

> For an overview of additional features available on Enterprise plans, see Pricing.

Why Use Cloud Deployment? (Advantages)

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

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

Getting Started

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

Your Organization will display a dedicated Portal, a network and access boundary that isolates your Cribl resources from all other users. Each Cribl.Cloud account provisions a separate AWS account. Your Cribl Stream instance is deployed inside a virtual private cloud (VPC) in this account.

The Portal will initially contain a free Cribl Stream instance. 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.

About Cribl Stream (and This Document)

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

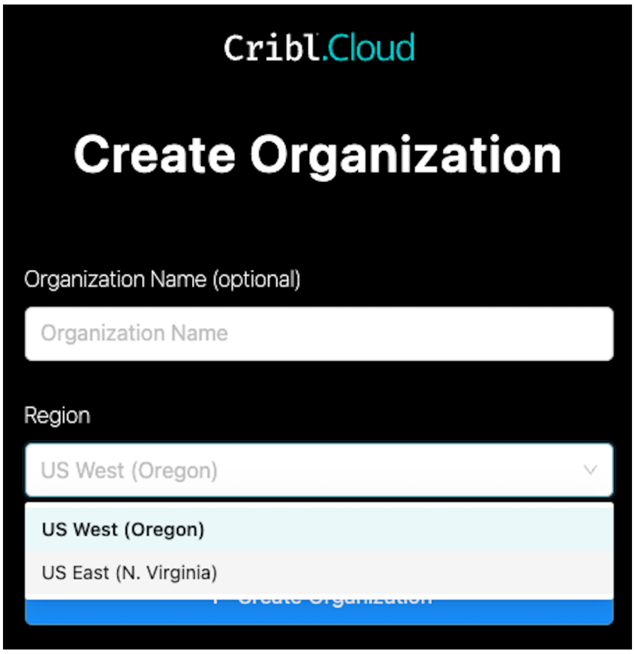
> Cribl.Cloud always runs in distributed mode – see Simplified Distributed Architecture below for details.

Registering a Cribl.Cloud Portal

Ready to take the red pill? The next few sections explain how to register and manage a Cribl.Cloud instance.

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

1. Start at: https://cribl.cloud/signup/
2. Select the **New User? Free signup** option, and register with your work email address.
3. Use the verification code from Cribl's email to confirm your registration.
4. On the **Create Organization** page, optionally enter an **Organization Name** (a friendly alias for the randomly generated ID that Cribl will assign to your Organization).
5. Select an AWS Region to host your Cribl.Cloud Leader and Cribl-managed Workers. Cribl currently supports either the `US West (Oregon)` or `US East (Virginia)` Region.
6. Bookmark your Cribl.Cloud portal page, for all that follows.


Cribl.Cloud

Create Organization

Organization Name (optional)

Organization Name

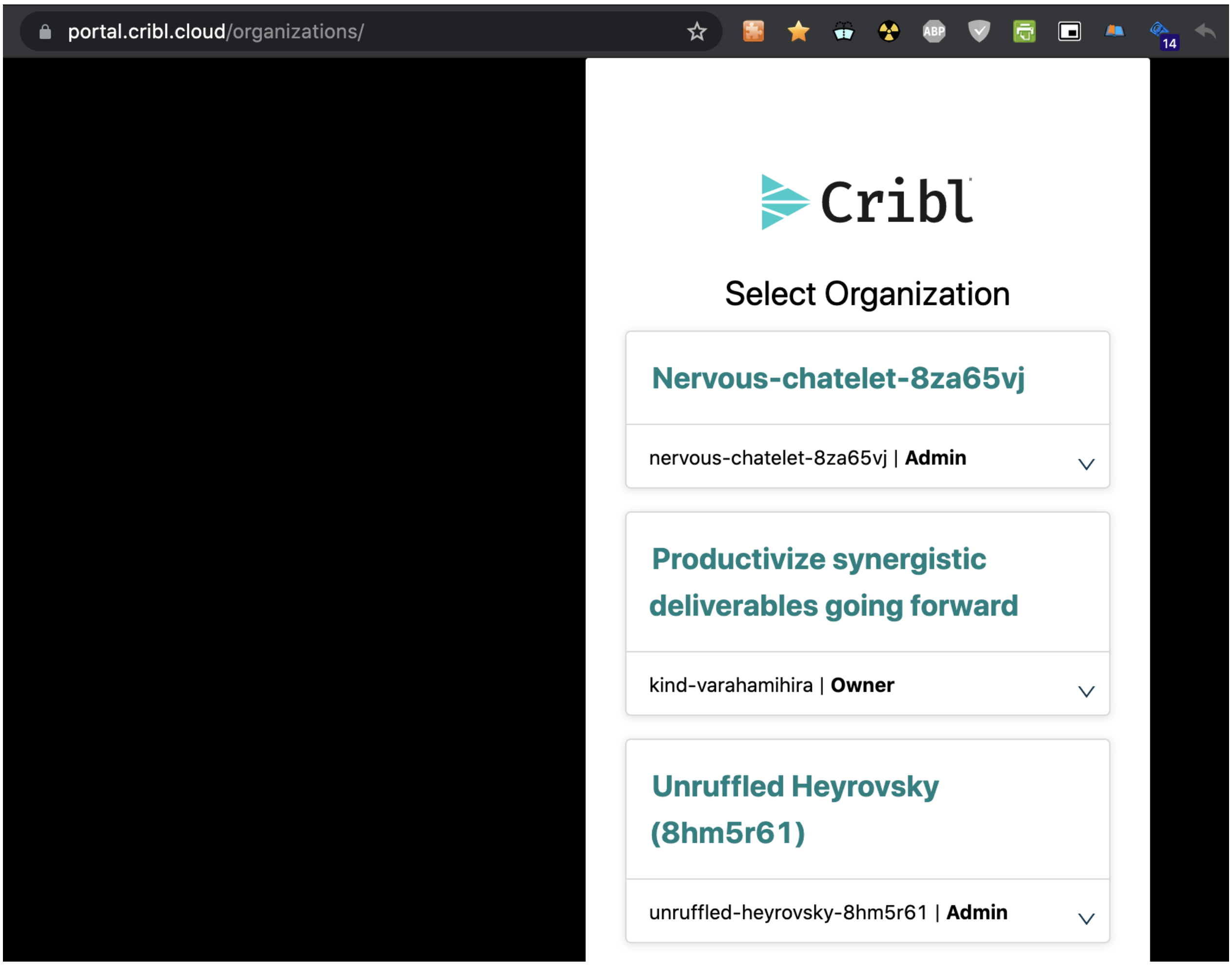
Region

US West (Oregon)

Create Organization page – selecting a host Region

Select Organization Page

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


portal.cribl.cloud/organizations/

Cribl

Select Organization

Nervous-chatelet-8za65vj

nervous-chatelet-8za65vj | **Admin**

Productivize synergistic deliverables going forward

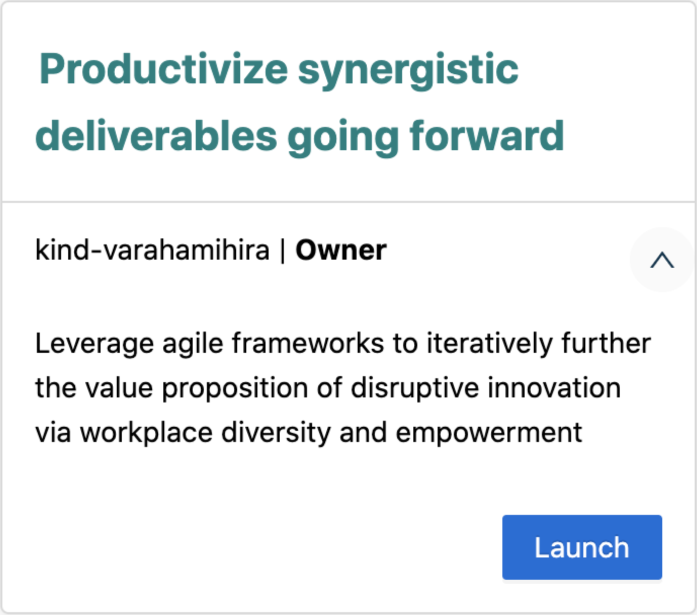
kind-varahamihira | **Owner**

Unruffled Heyrovsky (8hm5r61)

unruffled-heyrovsky-8hm5r61 | **Admin**

Select Organization interstitial page

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


Productivize synergistic deliverables going forward

kind-varahamihira | **Owner**

Leverage agile frameworks to iteratively further the value proposition of disruptive innovation via workplace diversity and empowerment

Organization tile's details and controls

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

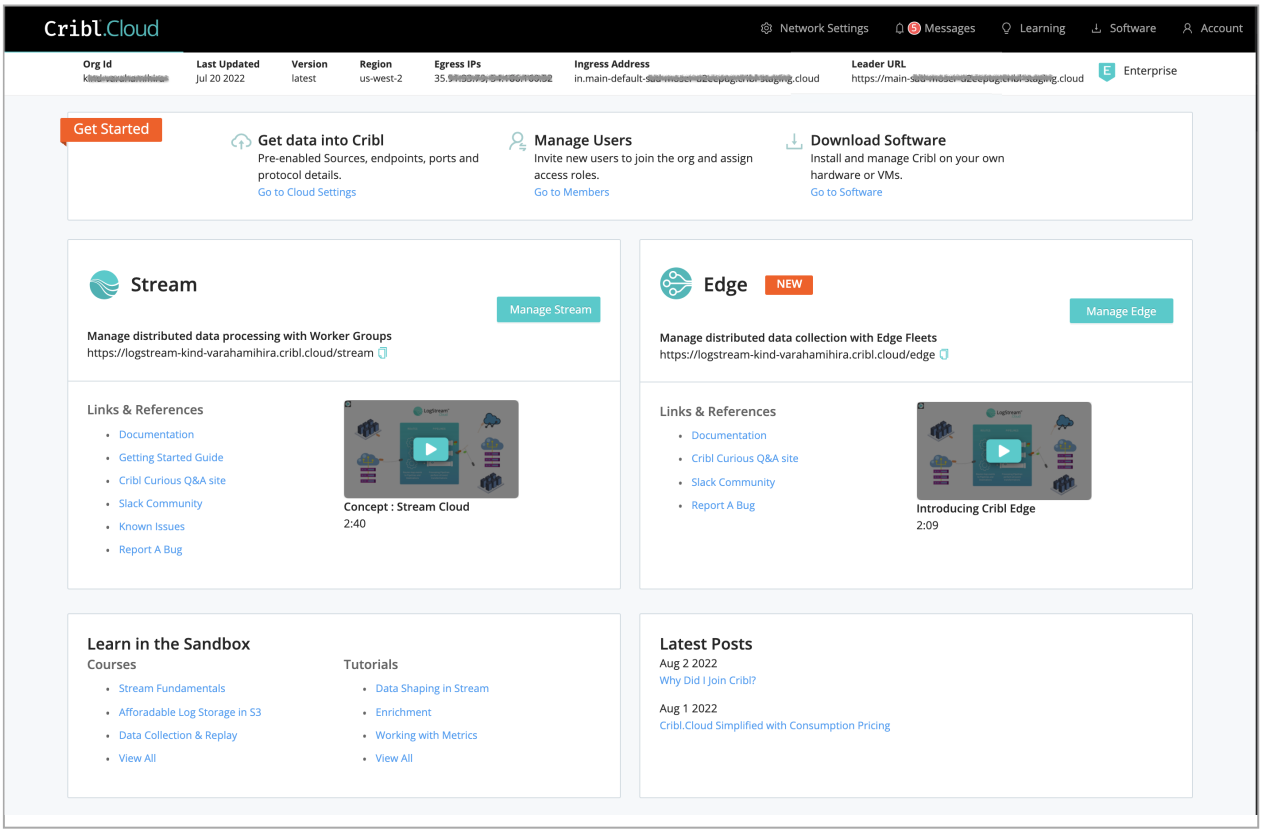
Exploring the Cribl.Cloud Portal

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

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

Portal Page (Cribl.Cloud Logo)

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



Org Id	Last Updated	Version	Region	Egress IPs	Ingress Address	Leader URL	
k[redacted]	Jul 20 2022	latest	us-west-2	35.[redacted]	in.main-default-[redacted]g.cloud	https://main-[redacted]g.cloud	Enterprise

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://logstream-kind-varahamihira.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 NEW

Manage distributed data collection with Edge Fleets
https://logstream-kind-varahamihira.cribl.cloud/edge

Links & References
- Documentation
- Cribl Curious Q&A site
- Slack Community
- Report A Bug

Introducing Cribl Edge
2:09

Learn in the Sandbox

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

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

Latest Posts
Aug 2 2022
Why Did I Join Cribl?

Aug 1 2022
Cribl.Cloud Simplified with Consumption Pricing

Cribl.Cloud portal

However, the surrounding page offers lots more useful information:

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

Overview Strip

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

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

Last Updated: Date on which Cribl last pushed an infrastructure change (notably including changes to the above **Egress Address**).

Version: Your deployed Cribl Stream version.

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

Egress IPs: Your Cribl.Cloud instance's current public IP address. This address is dynamic; Cribl will occasionally update it when we need to rescale core infrastructure.

Ingress Address: Your Cribl.Cloud instance's global domain for inbound data (before specifying ports per data type).

Network Settings Page

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

Data Sources Tab

The **Data Sources** tab lists ports, protocols, and data ingestion inputs that are open and available to use. Return to this tab to copy Ingest Addresses (endpoints) as needed. For details, see Available Ports and TLS Configurations.

ACL Tab

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.

Trust Tab

The **Trust** tab provides a **Worker ARN** (Amazon Resource Name) that you can copy and paste to attach a Trust Relationship to an AWS account's IAM role. Doing so 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.

Cribl Stream UI Access

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

Messages Drawer

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

Learning Page

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

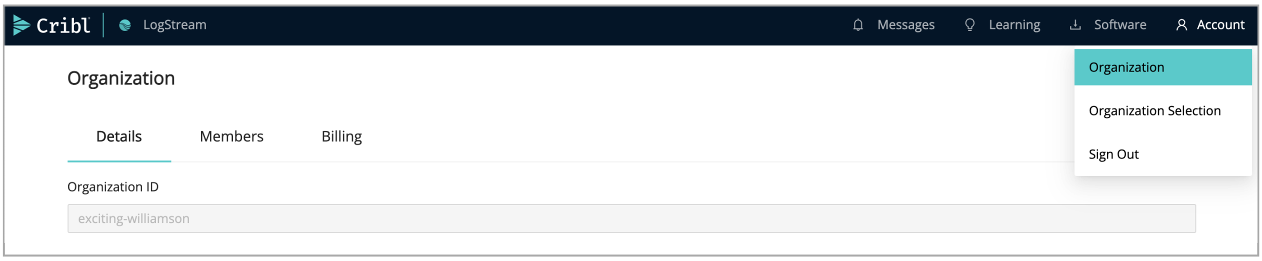
- Sandboxes (free, interactive tutorials on fully hosted integrations).
- Documentation.
- Product and plans overview (pricing comparison).
- Cribl events (including future and archived Webinars).
- Concept/demo videos.

Software Page

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

Account Menu

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



Organization ID: exciting-williamson

Account tab

Organization Page

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

Details Tab

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

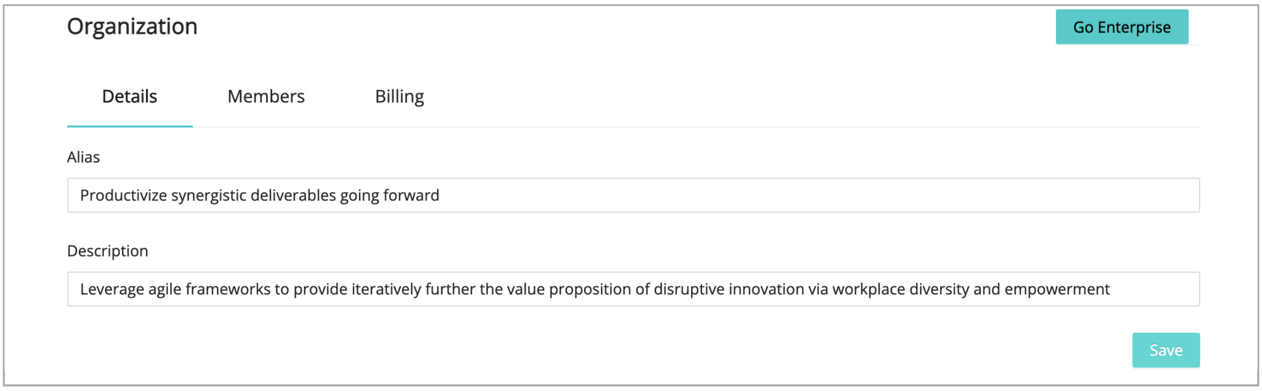
Alias: Optionally, enter a "friendly" name for your Organization. Upon signing in, members will see this alias above the Organization ID on the Select Organization page.

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

License: This drop-down displays one or multiple licenses that you've currently applied to 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 Cloud.

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



Alias: Productivize synergistic deliverables going forward

Description: Leverage agile frameworks to provide iteratively further the value proposition of disruptive innovation via workplace diversity and empowerment

Organization Details tab

Members Tab

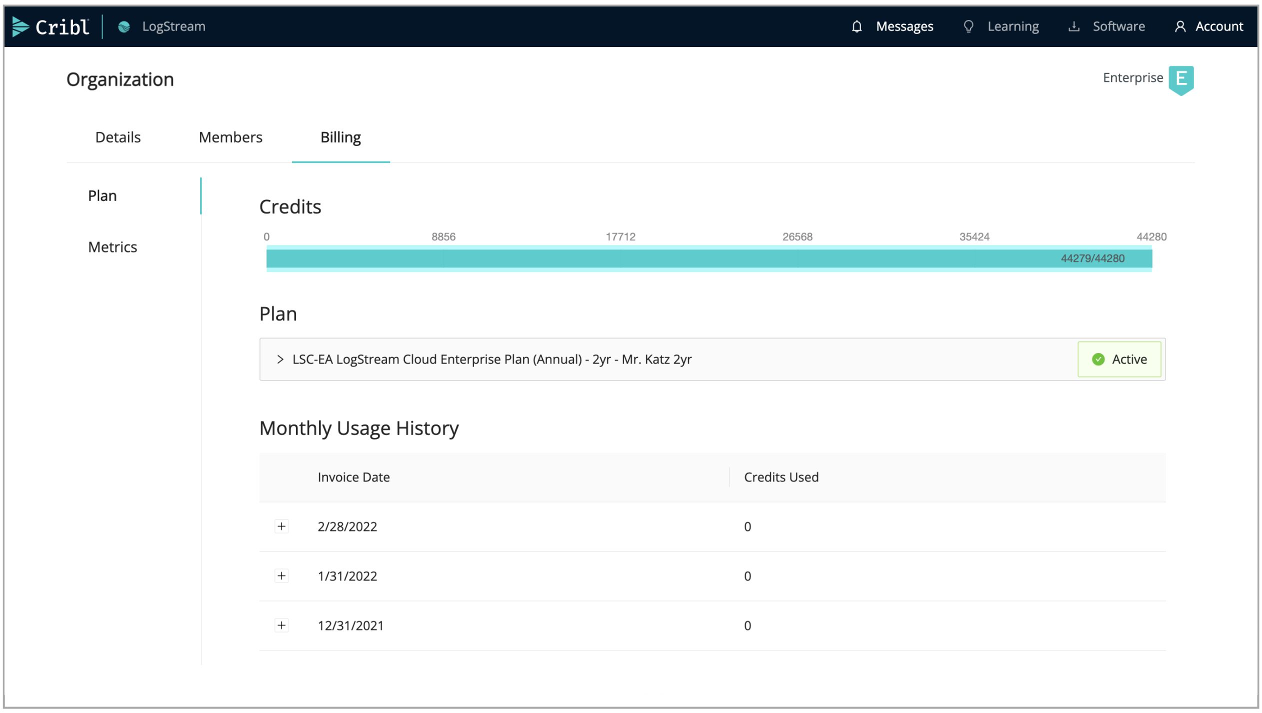
The **Organization** > **Members** upper tab provides access to inviting and managing other users.

Billing Tab

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

Plan Tab

The **Plan** left tab displays a mercury bar of available **Credits** on your account, an expandable **Plan** details accordion, and expandable **Monthly Usage History** rows offering details about your data throughput volume in current and prior months.



> LSC-EA LogStream Cloud Enterprise Plan (Annual) - 2yr - Mr. Katz 2yr   Active

Monthly Usage History

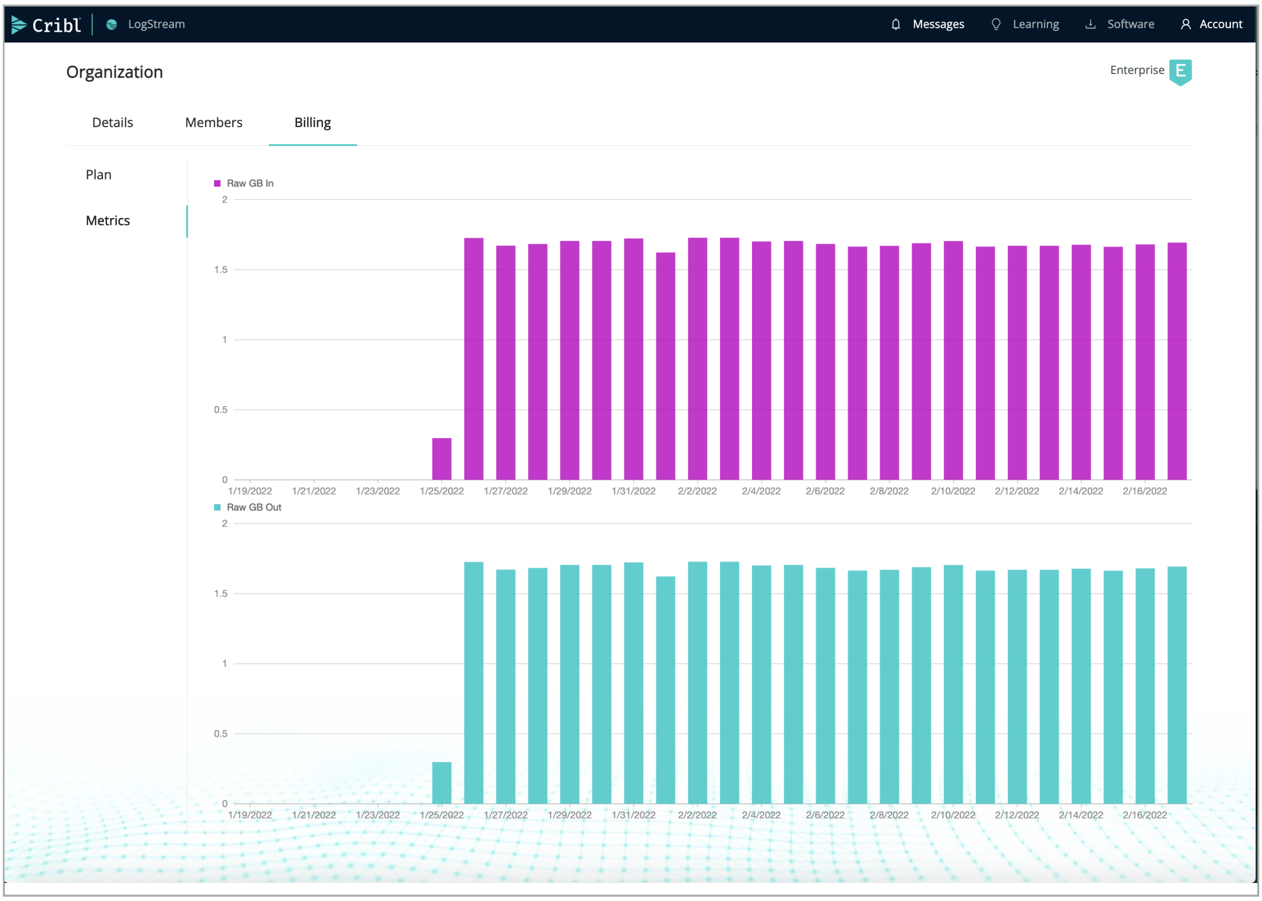
	Invoice Date	Credits Used
+	2/28/2022	0
+	1/31/2022	0
+	12/31/2021	0

Billing > Plan tab

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

Metrics Tab

The **Metrics** left tab provides bar graphs showing **Raw GB In** and **Raw GB Out** for each day over the last month.



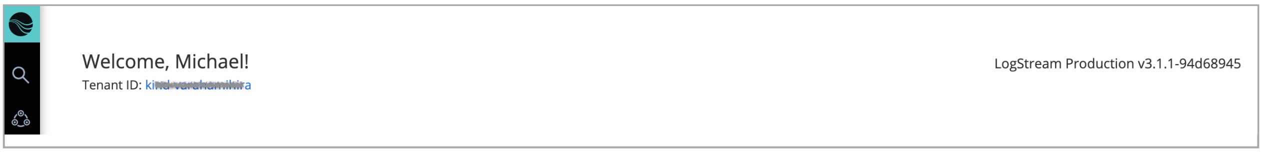
Billing > Metrics tab

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 either **Manage Stream** or **Manage Edge**.
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.



Welcome, Michael!

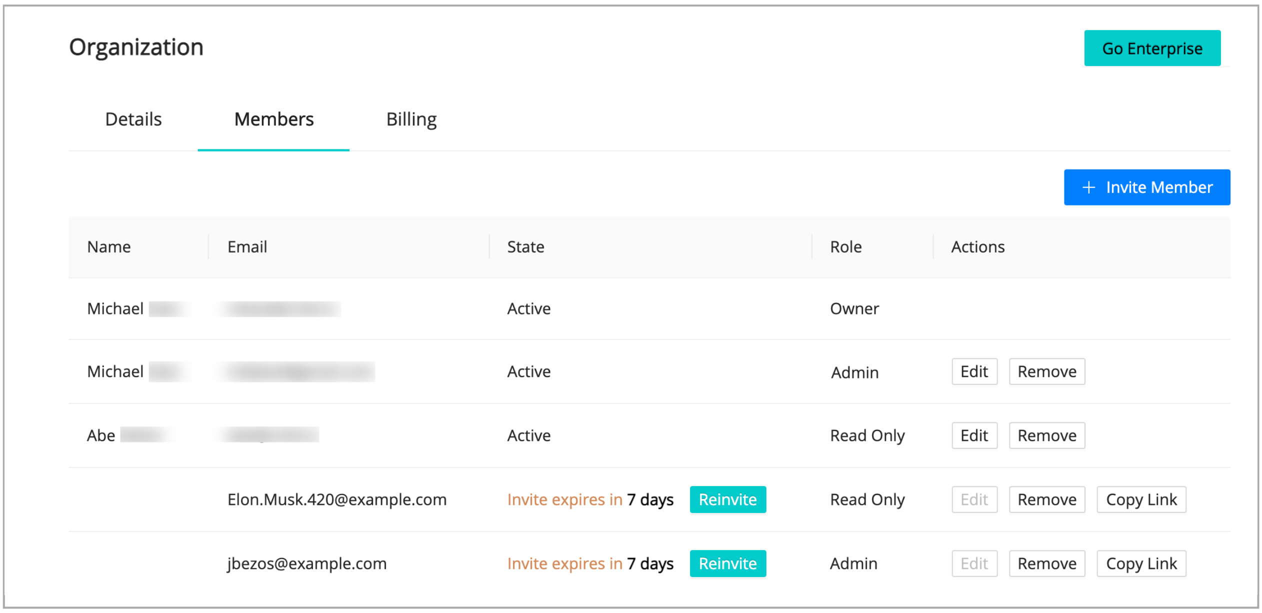
Tenant ID: [redacted]

LogStream Production v3.1.1-94d68945

Cribl.Cloud link takes you back to the portal

Inviting and Managing Other Users

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

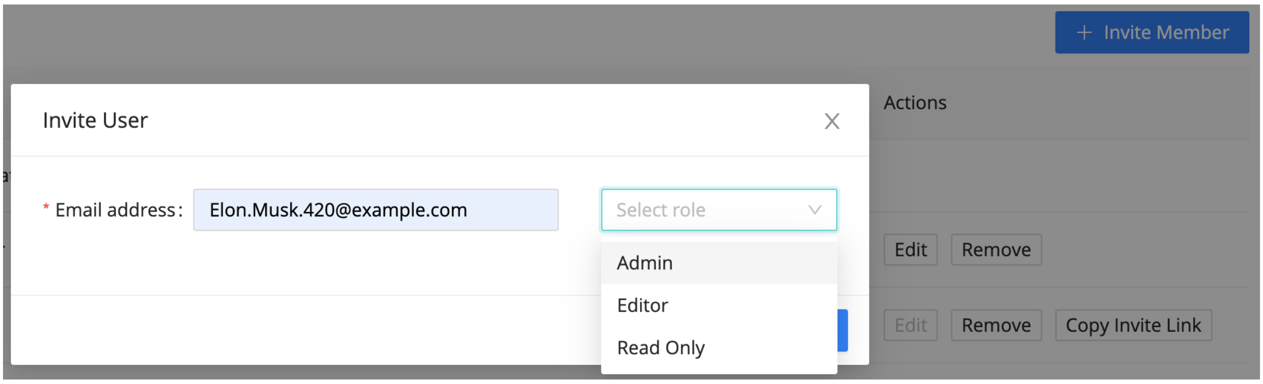


Name	Email	State	Role	Actions
Michael [redacted]	[redacted]	Active	Owner	
Michael [redacted]	[redacted]	Active	Admin	Edit Remove
Abe [redacted]	[redacted]	Active	Read Only	Edit Remove
	Elon.Musk.420@example.com	Invite expires in 7 days Reinvite	Read Only	Edit Remove Copy Link
	jbezos@example.com	Invite expires in 7 days Reinvite	Admin	Edit Remove Copy 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 a **Role** (explained just below), and then click **Invite** to send the invitation.



* Email address: Elon.Musk.420@example.com

Invite User modal

Member Roles

Each Role that you can assign to members confers a default Role within the Organization's Cribl.Cloud instance. Here are the Roles, their corresponding permissions, and who can assign each:

MEMBER ROLE	CRIBL STREAM ROLE	OPTIONS/RESTRICTIONS
`Admin`	`admin`	Any Organization owner can assign
`Editor`	`editor_all`	Assignable only with Enterprise plan
`Read-Only`	`reader_all`	Assignable only with Enterprise plan
`Owner`	`admin`	Can't be assigned, but can manage Organization details

Note that:

- Owners of non-Enterprise Cribl.Cloud Organizations can assign only the `Admin` Role in the **Invite User** modal shown above.
- Expanded role-based access control – i.e., the ability to manage the `Editor` and `Read-Only` Roles shown above – is available only with an Enterprise plan. (For all available Enterprise features, see Pricing.)
- The one Member Role that you cannot assign or transfer is your own Owner Role. A user can acquire this superuser Role only by signing up as the owner of their own Cribl.Cloud Organization.
- Only an Organization's Owner can manage the Organization's details.
- You assign Roles per individual user, when you invite them to your Organzation. Cribl.Cloud does not currently support globally predefining or assigning group Roles, as with on-premises Cribl Stream. However, Admins can change users' Roles after those users join their Organization.

Cribl.Cloud Roles Rule Cribl Stream Access

When you assign a Cribl.Cloud Member Role, it is mapped to a Cribl Stream Role as described above. However, these users will **not** be visible as local users within the UI of Cribl Stream Cloud instances managed by Cribl.

Also, within these instances' UI: modifying Roles **not** mapped above will have no effect; and adding local users wil have no effect.

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 Role level you've specified.

Managing Invites

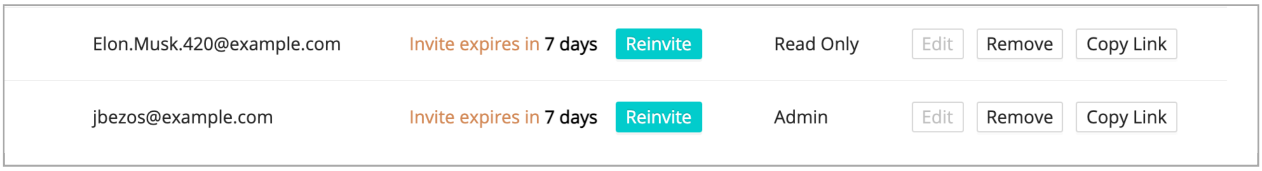
While an invite is pending, the **Organization** > **Members** tab offers you these options to deal with commonly encountered issues:

- **Reinvite**: If your invited member didn't receive your invitation email, you can click this button to resend it.

- **Copy Link**: If emails aren't getting through at all, click this button to copy and share a URL that will take the invitee directly to the signup page. This target page encapsulates the same identity, Organization, and Role you specified in the original email invite.

- **Remove**: This is for scenarios where you need to revoke a pending invite. (You sent someone a duplicate invite, your invitee is spending too much time in space to be a productive collaborator, etc.) After clicking this button, you'll see a confirmation dialog.

After 7 days, if an invite has been neither accepted nor revoked, it expires. In this case, it is removed from the **Members** tab.



Elon.Musk.420@example.com	Invite expires in 7 days	Reinvite	Read Only	Edit	Remove	Copy Link
jbezos@example.com	Invite expires in 7 days	Reinvite	Admin	Edit	Remove	Copy Link

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 Role. (The **Edit** option is displayed only if you have an Enterprise plan.)

- **Remove**: Remove this member from your Organization. After clicking this button, you'll see a confirmation dialog. (Proceeding will not affect this user's access to any other Cribl.Cloud Organizations they might own or be members of.)

Cloud Pricing

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

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

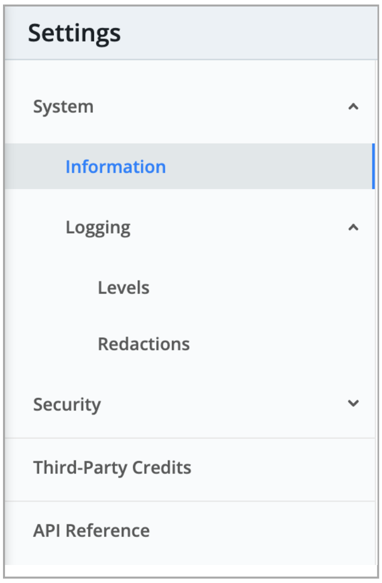
Differences from Self-Hosted Cribl Stream

A Cribl.Cloud deployment can differ from an on-premises/customer-managed Cribl Stream deployment in the following ways. 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 left nav and Settings page.



Cribl.Cloud's 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. With a Free or Standard plan, there is a single Worker button and Worker.

Compared to self-hosted Cribl Stream, the **Settings** > **Worker Processes** and **Settings** > **Distributed Settings** links are omitted, and the left nav contains no **Worker Groups** or **Mappings** links.

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 Workers and Worker Groups.

Git Preconfigured

Without an Enterprise plan, the **Settings** > **System** > **Git Settings** section is omitted. A local `git` client is preconfigured in your Cribl.Cloud portal. On Cribl.Cloud's left nav, use the **Changes** link to commit/push changes to `git`. Select **Deploy** at the UI's top right 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

Without an Enterprise plan, the **Settings** > **Access Management** section is omitted. All users of a given Cribl.Cloud instance share a single `admin` login.

Without an Enterprise plan, the **Settings** > **Security** > **KMS** section is omitted. The free version does not support KMS secrets stores.

If you add an Enterprise Plan, Cloud and hybrid Leaders support Local and Google SSO authentication. However, they do not currently support other OpenID Connect authentication schemes, nor LDAP, nor the SAML protocol.

Role-based access control (RBAC) is simplified in Cribl.Cloud. For details, see Member Roles.

Transparent Licensing

The left nav's **Settings** > **Licensing** link is omitted. Your license is managed by your parent Cribl.Cloud portal, where you can check credits and usage history on the Billing tab.

Other Simplified Settings

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

- The Script Collector, and custom Functions, are available only on hybrid, customer-managed Workers. (These features are currently not available on Cribl-managed Workers.)
- The left nav's **Settings** > **Scripts** link is omitted from Cribl.Cloud, which currently does not support configuring or running shell scripts on hybrid or Cribl-managed Workers.
- The Filesystem Collector and Filesystem Destination are available only on hybrid Workers. (Cribl-managed Workers have no local filesystem to read from or write to.)
- Persistent Queues can be configured only on hybrid (not Cribl-managed) Workers' Sources and Destinations.
- File-based Destinations support staging directories only on hybrid (not Cribl-managed) Workers.
- The Tee Function is available only on hybrid (not Cribl-managed) Workers.

Support Options

The **Settings** > **Diagnostics** link is omitted. For help with any troubleshooting needs:

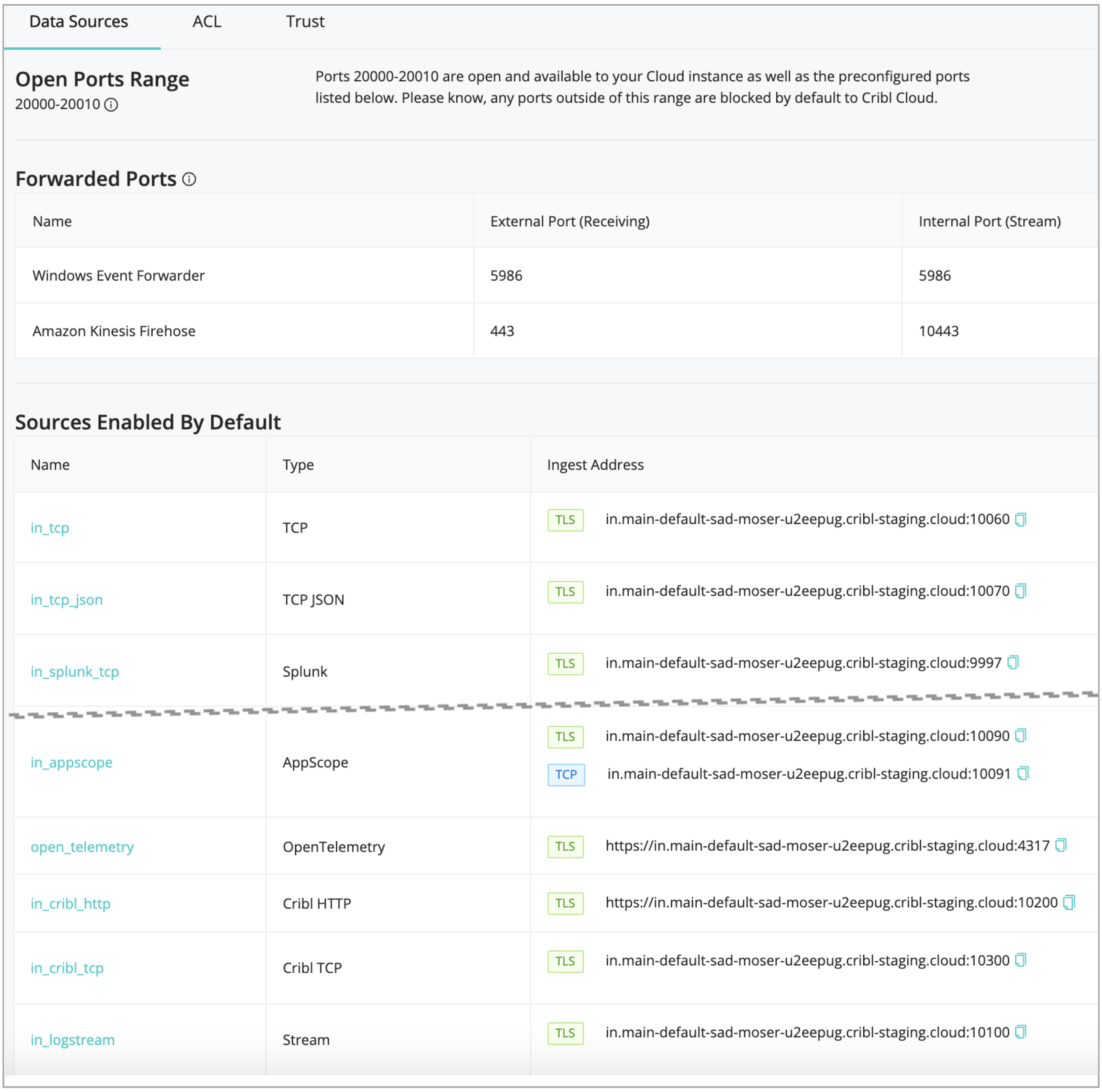
- Click the Intercom link at Cribl Stream's lower right.
- Join Cribl's Community Slack `#cribl-cloud` channel.

- If you have a paid account, contact Cribl Support.

Available Ports and TLS Configurations

To get data into Cribl.Cloud, your Cribl.Cloud portal provides several data sources and ports already enabled for you, plus 11 additional ports (`20000-20010`) that you can use to add and configure more Cribl Stream 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 the existing Sources listed here, Cribl recommends using the preconfigured endpoint and port to send data into Cribl Stream.



Open Ports Range
20000-20010

Ports 20000-20010 are open and available to your Cloud instance as well as the preconfigured ports listed below. Please know, any ports outside of this range are blocked by default to Cribl Cloud.

Forwarded Ports

Name	External Port (Receiving)	Internal Port (Stream)
Windows Event Forwarder	5986	5986
Amazon Kinesis Firehose	443	10443

Sources Enabled By Default

Name	Type	Ingest Address
in_tcp	TCP	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10060
in_tcp_json	TCP JSON	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10070
in_splunk_tcp	Splunk	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:9997
in_appscope	AppScope	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10090 TCP in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10091
open_telemetry	OpenTelemetry	TLS https://in.main-default-sad-moser-u2eepug.cribl-staging.cloud:4317
in_cribl_http	Cribl HTTP	TLS https://in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10200
in_cribl_tcp	Cribl TCP	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10300
in_logstream	Stream	TLS in.main-default-sad-moser-u2eepug.cribl-staging.cloud:10100

Available ports and TLS certificates

TLS Details

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

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

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

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

> **Cribl HTTP and Cribl TCP Sources/Destinations**
>
> Use the Cribl HTTP Destination and Source, and/or the Cribl TCP Destination and Source, to relay data between Worker Nodes connected to the same Leader. This traffic does not count against your ingestion quota, so this routing 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 Cribl Internal Source is omitted from Cribl.Cloud instances, because Cribl manages these instances' uptime and diagnostics on your behalf. Also, the Exec Source, available in self-hosted v.3.3 and above, is unavailable in Cribl.Cloud instances.

> 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.

Enterprise Cloud

With an Enterprise plan, Cribl.Cloud offers the same options and flexibility as a customer-managed Cribl Stream distributed deployment with an Enterprise license – and more.

These options include configuring and managing multiple Worker Groups or Fleets,maintaining version control with remote repos, Notifications, Google SSO authentication, and Role-based access control to Cribl Stream resources.

> For other Enterprise features, see Pricing.

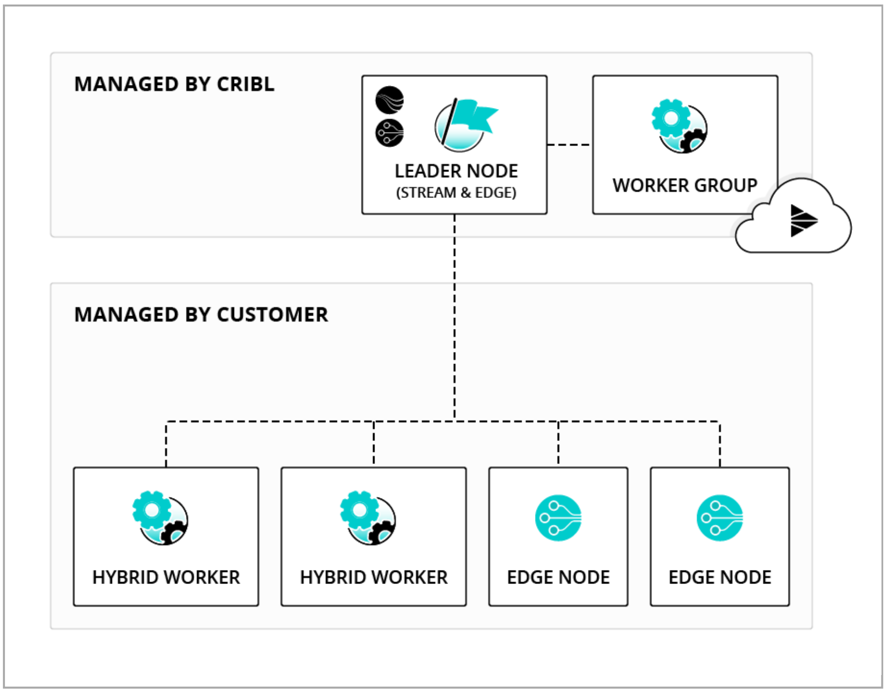
Cribl.Cloud Enterprise also adds:

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

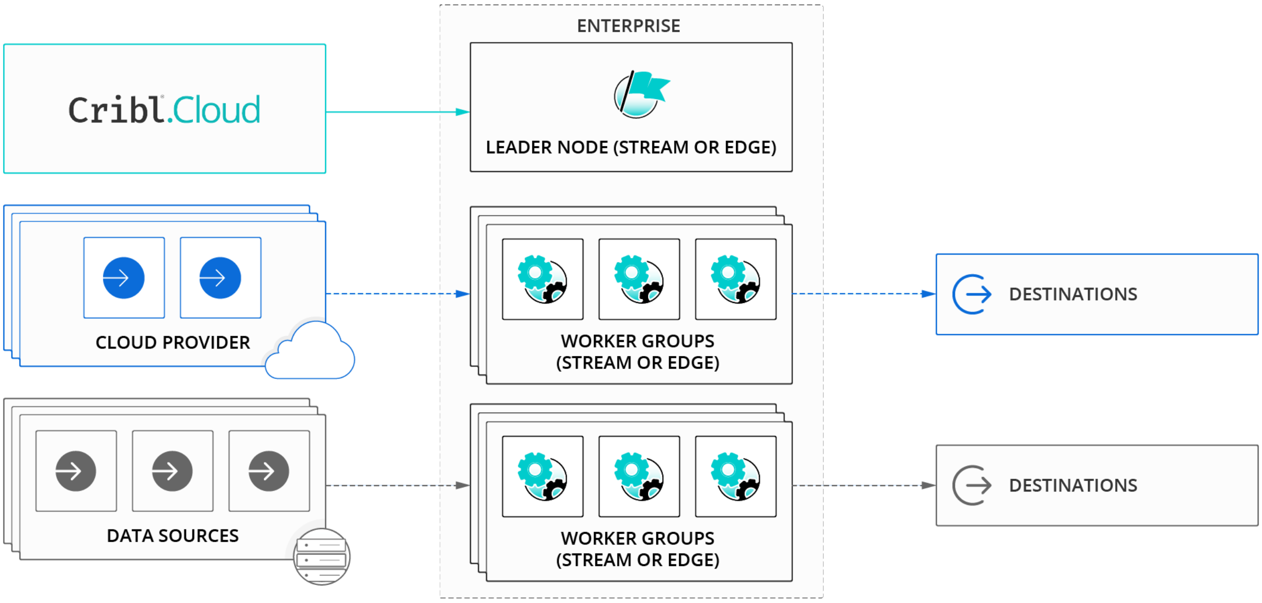
Hybrid Deployment

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

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



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



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

As the footprint of your operations grows or changes, this flexibility makes it easy to reconfigure Cribl 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, and reassign them to different Worker Groups, by easily auto-generating 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-premises, 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 Workers.
- On all Workers' hosts, port 4200 must be open for management by the Leader.
- On all Workers' hosts, firewalls must allow outbound communication on port 443 to the Cribl.Cloud Leader, and on port 443 to https://cdn.cribl.io.
- If this traffic must go through a proxy, see System Proxy Configuration for configuration details.

Note that you are responsible for data encryption and other security measures on Worker instances that you manage.

Adding (Bootstrapping) Workers

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

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

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

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

Hybrid Cribl HTTP/Cribl TCP Configuration

If you use the Cribl HTTP Destination and Source pair, or the Cribl TCP Destination and Source pair, to relay data between 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.

- Configure hybrid Workers by logging directly into their UI, then selecting global ⚙ **Settings** (lower left) > **Distributed Settings**. Make sure the **Mode** is set to **Managed Worker** or **Managed Edge** (which might require a restart).
- Then select the **Leader Settings** left tab, and ensure a consistent entry in the **Address** field.
- In Cloud hybrid deployments, the Leader's Address format is `main-<your-Org-ID>.cribl.cloud.` When configuring a hybrid Worker, use that format in the **Address** field.

;

3. Deploying Cribl Stream Software

3.1. 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.

Type of Deployment

- Use Cribl Cloud to quickly launch a Cribl-hosted Cribl Stream instance, with all infrastructure and management responsibility delegated to Cribl.
- Use Single-Instance 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.

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.

REQUIREMENT TYPE	REQUIREMENTS DETAILS
Minimum Leader and Worker Nodes.	**OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse (64bit) **System**: +4 physical cores, +8GB RAM, 5GB free disk space (more if persistent queuing is enabled on Workers)
Recommended Leader Node	**OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse (64bit) **System**: +4 physical cores, +8GB RAM, 5GB free disk space
Recommended Worker Nodes	**OS**: Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse (64bit) **System**: +8 physical cores, +32GB RAM, 5GB free disk space.

Browser Requirements

Most modern browsers will work, but here are the minimum supported versions as of this publication date: Firefox 65+, Chrome 70+, Safari 12+, Microsoft Edge.

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 initial ports listed below:

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

- Download, Install, and Launch Cribl.

- Enable Start at Boot.

- Configure as a Leader.

- Confirm Worker Processes Settings at `-2` (via global ⚙ **Settings** (lower left) > **System** > **Manage Processes**).

- Install License.

3. Configure Worker Nodes

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

DEFAULT PORT	PURPOSE
TCP:9000	Cribl UI

  - 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 global ⚙ **Settings** (lower left) > **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.

;

3.2. 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.

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 global ⚙ **Settings** (lower left) > **System** > **Worker Processes** section.

In a distributed deployment, you allocate resources per Worker Group. Navigate to **Groups >** `group-name` > **Settings** (upper right) > **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 memory available to each Worker Process, in MB. Defaults to `2048`. (See Estimating Memory Requirements below.)

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 global ⚙ **Settings** (lower left) > **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).

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.**
   E.g., it's better to have 5 connections to TCP 514, each bringing in 200GB/day, than one at 1TB/day.

2. Each Worker Process will maintain and manage its own outputs. E.g., if an instance with 2 Worker Processes is configured with a Splunk output, then the Splunk destination will see 2 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:

- Per 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.
- Per Node, Cribl recommends no more than 48 ARM vCPUs, or 48 x86 vCPUs (i.e., 24 cores with hyperthreading). This upper limit simply avoids wasting vCPU capacity due to running out of available sockets. It also 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)

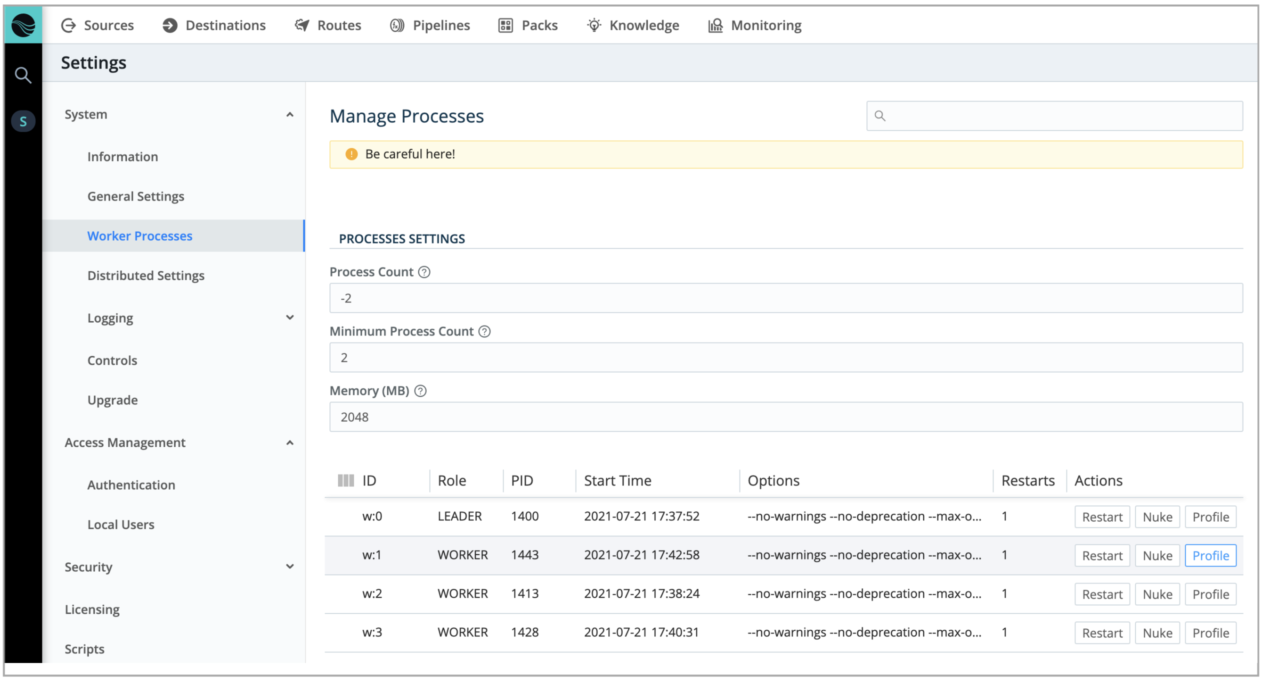
In all cases, reserve at least 5GB disk storage per instance, and more if persistent queuing is enabled.

Measuring CPU Load

You can profile CPU usage on individual Worker Processes.

Single-Instance Deployment

Go to global ⚙ **Settings** (lower left) > **System** > **Worker Processes**, and click **Profile** on the desired row.



Manage Processes

Be careful here!

PROCESSES SETTINGS

Process Count: -2

Minimum Process Count: 2

Memory (MB): 2048

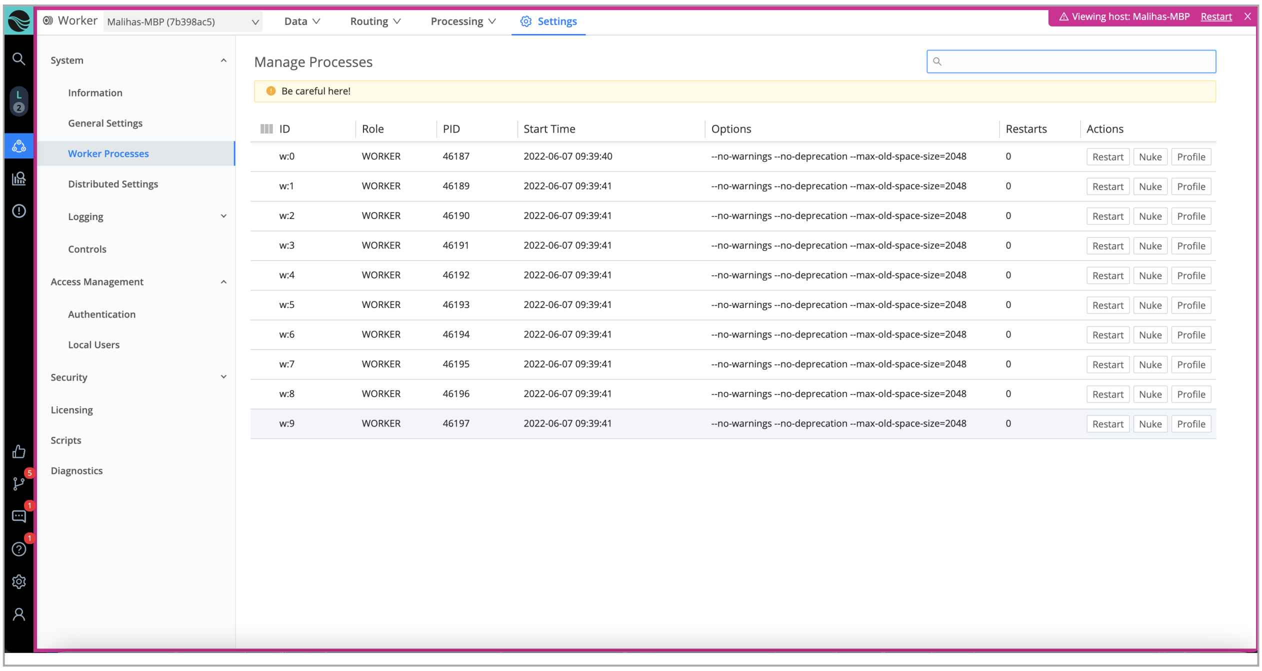
ID	Role	PID	Start Time	Options	Restarts	Actions
w:0	LEADER	1400	2021-07-21 17:37:52	--no-warnings --no-deprecation --max-o...	1	Restart Nuke Profile
w:1	WORKER	1443	2021-07-21 17:42:58	--no-warnings --no-deprecation --max-o...	1	Restart Nuke Profile
w:2	WORKER	1413	2021-07-21 17:38:24	--no-warnings --no-deprecation --max-o...	1	Restart Nuke Profile
w:3	WORKER	1428	2021-07-21 17:40:31	--no-warnings --no-deprecation --max-o...	1	Restart Nuke 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. Select **Manage** in the left nav.
3. On the resulting **Manage Groups** page, click the **Workers** tab.
4. 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.)
5. Select **Settings** from that Worker Node's top nav.
6. Select **System** > **Worker Processes** from the resulting side nav.
7. Click **Profile** on the desired Worker Process.



Manage Processes

Be careful here!

ID	Role	PID	Start Time	Options	Restarts	Actions
w:0	WORKER	46187	2022-06-07 09:39:40	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:1	WORKER	46189	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:2	WORKER	46190	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:3	WORKER	46191	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:4	WORKER	46192	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:5	WORKER	46193	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:6	WORKER	46194	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:7	WORKER	46195	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:8	WORKER	46196	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile
w:9	WORKER	46197	2022-06-07 09:39:41	--no-warnings --no-deprecation --max-old-space-size=2048	0	Restart Nuke Profile

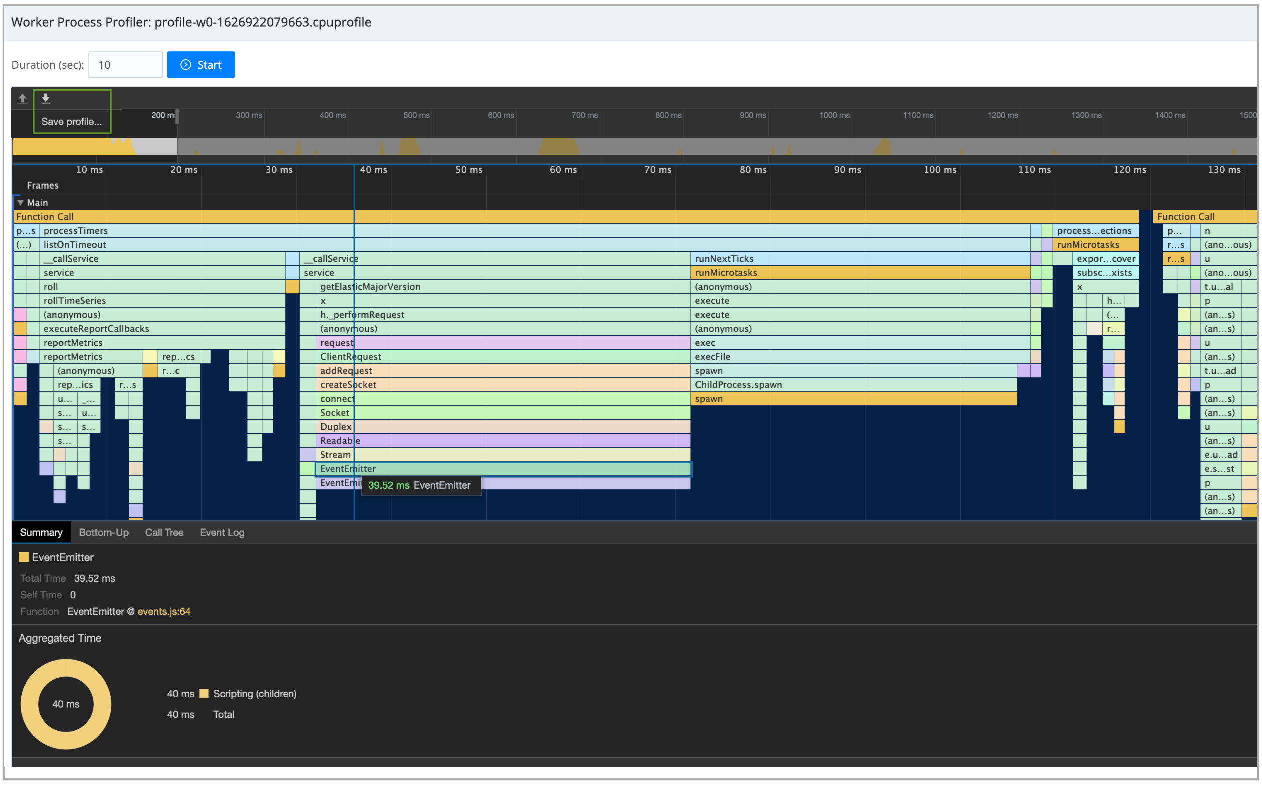
Worker CPU profiling (distributed)

Generating a CPU 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 Process Profiler: profile-w0-1626922079663.cpuprofile

Duration (sec): 10

EventEmitter

Total Time 39.52 ms

Self Time 0

Function EventEmitter @ events.js:64

Aggregated Time

40 ms 🟨 Scripting (children)

40 ms Total

Worker CPU 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 very small tiny minuscule **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.

;

3.3. Manual Deployment

3.3.1. 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.

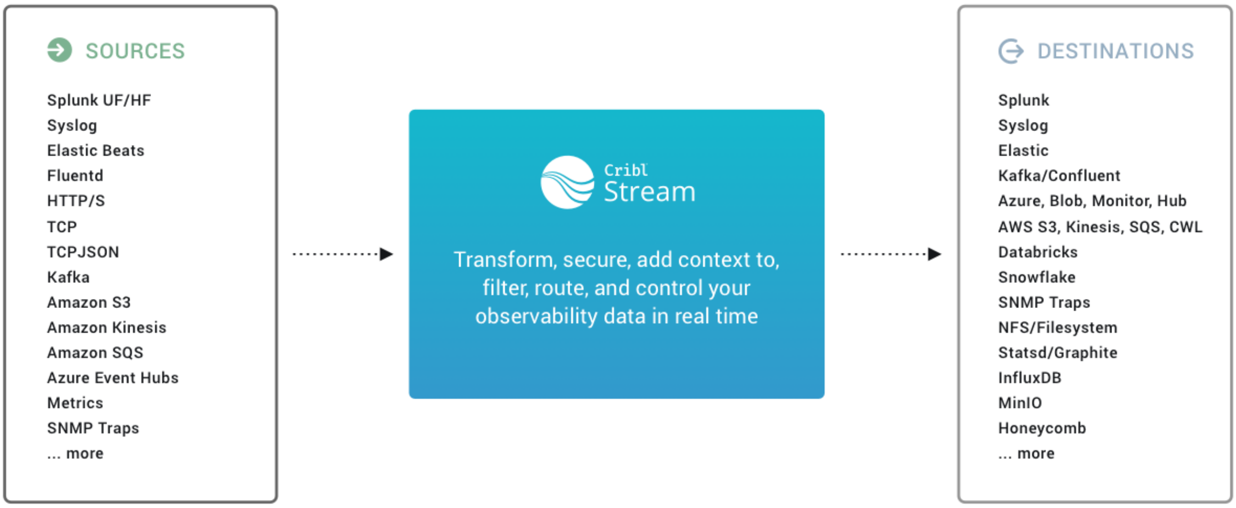
;

3.3.2. Single-Instance 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.

Architecture


SOURCES

Splunk UF/HF
Syslog
Elastic Beats
Fluentd
HTTP/S
TCP
TCPJSON
Kafka
Amazon S3
Amazon Kinesis
Amazon SQS
Azure Event Hubs
Metrics
SNMP Traps
... more

Cribl Stream

Transform, secure, add context to, filter, route, and control your observability data in real time

DESTINATIONS

Splunk
Syslog
Elastic
Kafka/Confluent
Azure, Blob, Monitor, Hub
AWS S3, Kinesis, SQS, CWL
Databricks
Snowflake
SNMP Traps
NFS/Filesystem
Statsd/Graphite
InfluxDB
MinIO
Honeycomb
... more

Requirements

- **OS (Intel Processors):**
    - Linux 64-bit kernel >= 3.10 and glibc >= 2.17
    - Examples: Ubuntu 16.04+, Debian 9+, RHEL 7+, CentOS 7+, SUSE Linux Enterprise Server 12+, Amazon Linux 2014.03+
- **OS (ARM64 Processors):**
    - Linux 64-bit
    - Tested so far on Ubuntu (14.04, 16.04, 18.04, and 20.04), CentOS 7.9, and Amazon Linux 2

- **System:**
  - +4 physical cores, +8GB RAM
  - 5GB 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 Support**: Firefox 65+, Chrome 70+, Safari 12+, Microsoft Edge
- **SELinux Support**: `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.

Network Ports

By default, Cribl Stream listens on the following ports:

COMPONENT	DEFAULT PORT
UI	`9000`
HTTP In	`10080`
Splunk to Cribl Stream data port	`localhost:10000` (Cribl App for Splunk)
`| criblstream` Splunk search command to Cribl Stream	`localhost:10420` (Cribl App for Splunk)
User options	+ Other data ports as required.

Overriding Default Ports

The above ports can be overridden in the following configuration files:

- Cribl UI port (`9000`): Default definitions for `host`, `port`, and other settings are set in `$CRIBL_HOME/default/cribl/cribl.yml`, and can be overridden by defining alternatives in `$CRIBL_HOME/local/cribl/cribl.yml`.
- Data Ports: HTTP In (`10080`), TCPJSON in (`10420`) Splunk to Cribl (`10000`): Default definitions for `host`, `port` and other settings are set in `$CRIBL_HOME/default/cribl/inputs.yml`, and can be overridden by defining alternatives in `$CRIBL_HOME/local/cribl/inputs.yml`.

> **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.

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 `/opt/`:
  - `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

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

Using systemd

To **enable** Cribl 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.

```
[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 is required to listen on 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
> ```

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.

;

3.3.3. 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.

To sustain high availability, failover to an alternate Leader is available in Cribl Stream 3.5 and above.

> See common distributed deployment use cases in Worker Groups – What Are They and Why You Should Care.
>
> 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 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

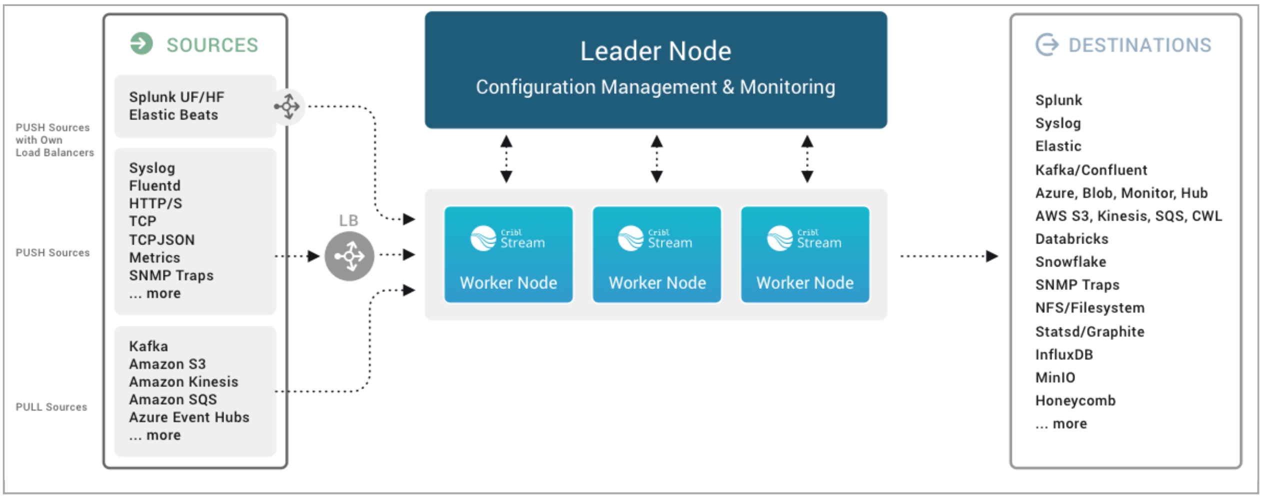
To clarify how the above concepts add up hierarchically, let's use a military metaphor involving toy soldiers:

- 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.

Leader Node Requirements

- **OS**:

  - Linux: RedHat, CentOS, Ubuntu, AWS Linux (64bit).

- **System**:

  - +4 physical cores, +8GB RAM.
  - 5GB free disk space.

- **Git**: `git` must be available on the Leader Node. See details below.
- **Browser Support**: Firefox 65+, Chrome 70+, Safari 12+, Microsoft Edge

> We assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs). All quantities listed above are minimum requirements.
>
> We recommend deploying the Leader on stable, highly available infrastructure, because of its role in coordinating all 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. The subsections below list these separately for the Leader versus Workers.

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.

Network Ports – Leader Node

In a distributed deployment, Workers communicate with the Leader Node on these ports. Ensure that the Leader is reachable on those ports from **all** Workers.

COMPONENT	DEFAULT PORT	PROTOCOL
Cluster communications	`4200`	TCP (Raw)
Config deployment	`4200`	HTTP/S

COMPONENT	DEFAULT PORT	PROTOCOL
Bootstrap (optional)	`9000`	HTTP/S

In addition to the existing cluster connection, which is persistent, a separate HTTP connection is established for config deployments from Workers to the Leader on `TCP:4200`. This lasts connection only until the config is downloaded.

> Cluster communication cannot go through proxies because this communication is not HTTP-based.

Network Ports – Worker Nodes

By default, all Cribl Stream Worker instances listen on the following ports:

COMPONENT	DEFAULT PORT
UI	`9000`
User options	+ Other data ports as required.

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

In global ⚙ **Settings** (lower left) > **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.

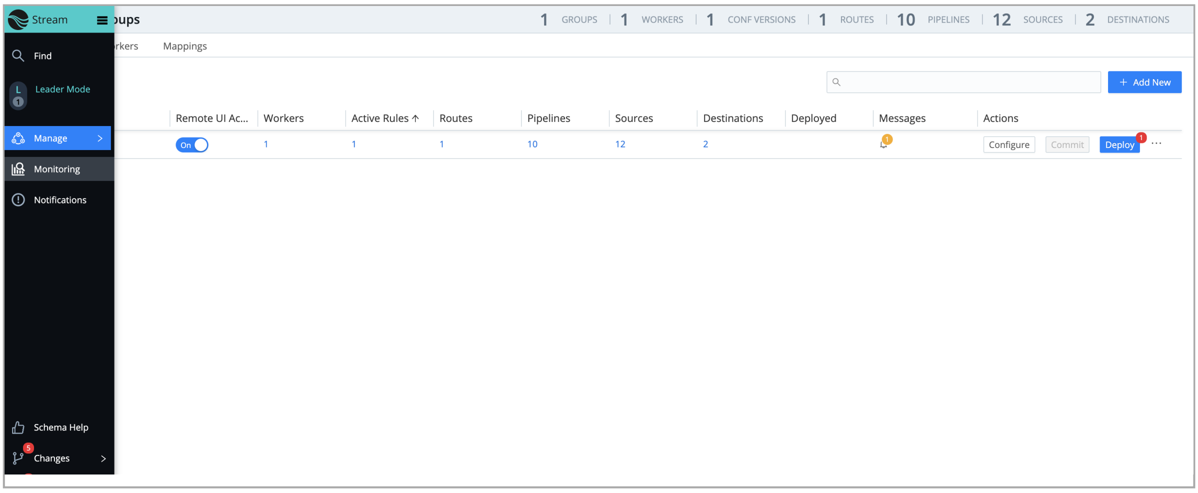
Remote UI Access

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 left nav, click **Manage**.

2. On the **Manage Groups** page: For each desired Worker Group, toggle **Remote UI Access** to `On`.



Remote UI Ac...	Workers	Active Rules ↑	Routes	Pipelines	Sources	Destinations	Deployed	Messages	Actions
On	1	1	1	10	12	2			Configure Commit Deploy ···

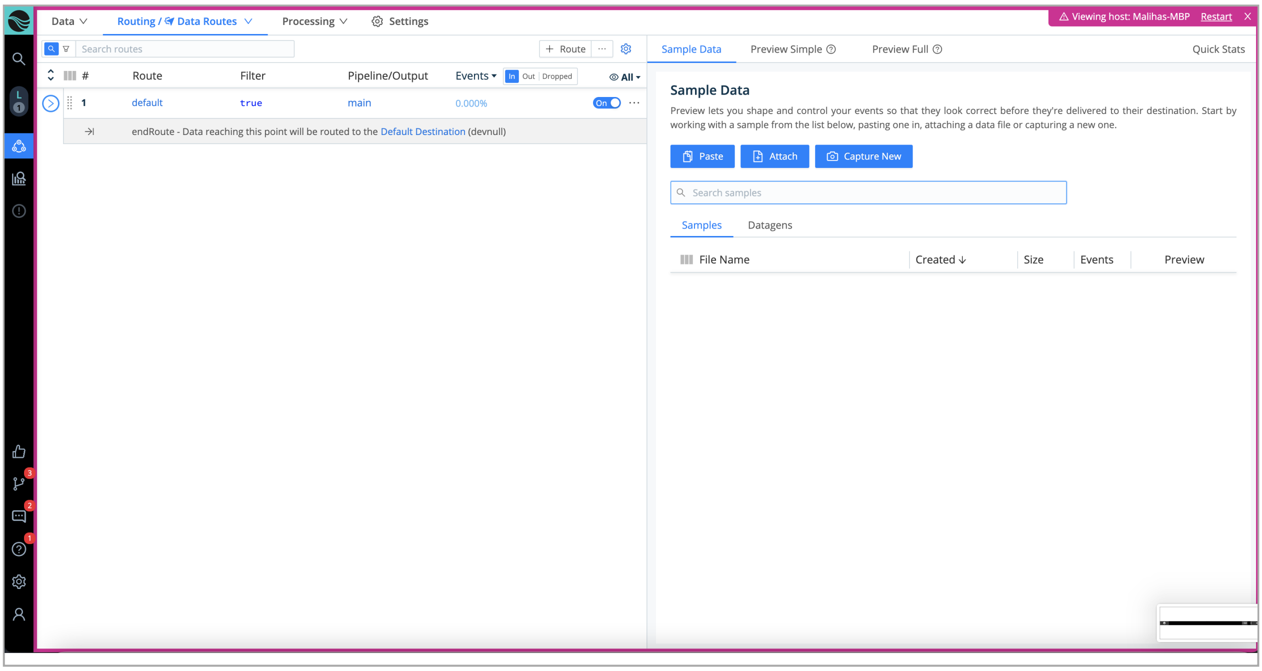
Worker UI access setting

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 purple border, with a badge labeled **Viewing host:** `<host/GUID>`.

   At the upper right, you can click **Restart** to restart the Worker, or click the close box to return to the **Manage Workers** page. Note that any changes you make here will not be propagated to the Leader.



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

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.

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.

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.

Using the UI

In global ⚙ **Settings** (lower left) > **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., `cribleader.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://criblmaster@masterHostname:4203 ./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]
```

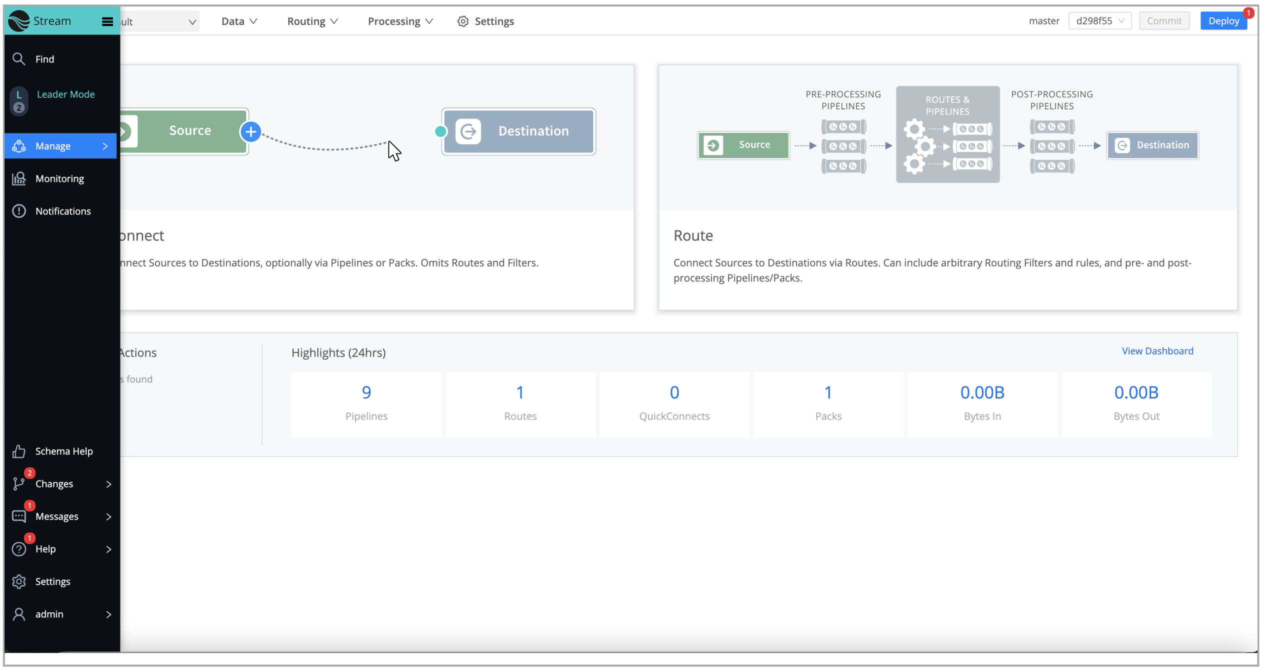
The `-H` and `-p` parameters are required. For other options, see the CLI Reference. Here is an example command:

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

Cribl Stream will need to restart after this command is issued.

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. The left nav adds **Leader Mode**, and **Manage** tabs – all to manage Workers and their assignments. Also, the global **Monitoring** link moves from the top to the left nav.



onnect

nnect Sources to Destinations, optionally via Pipelines or Packs. Omits Routes and Filters.

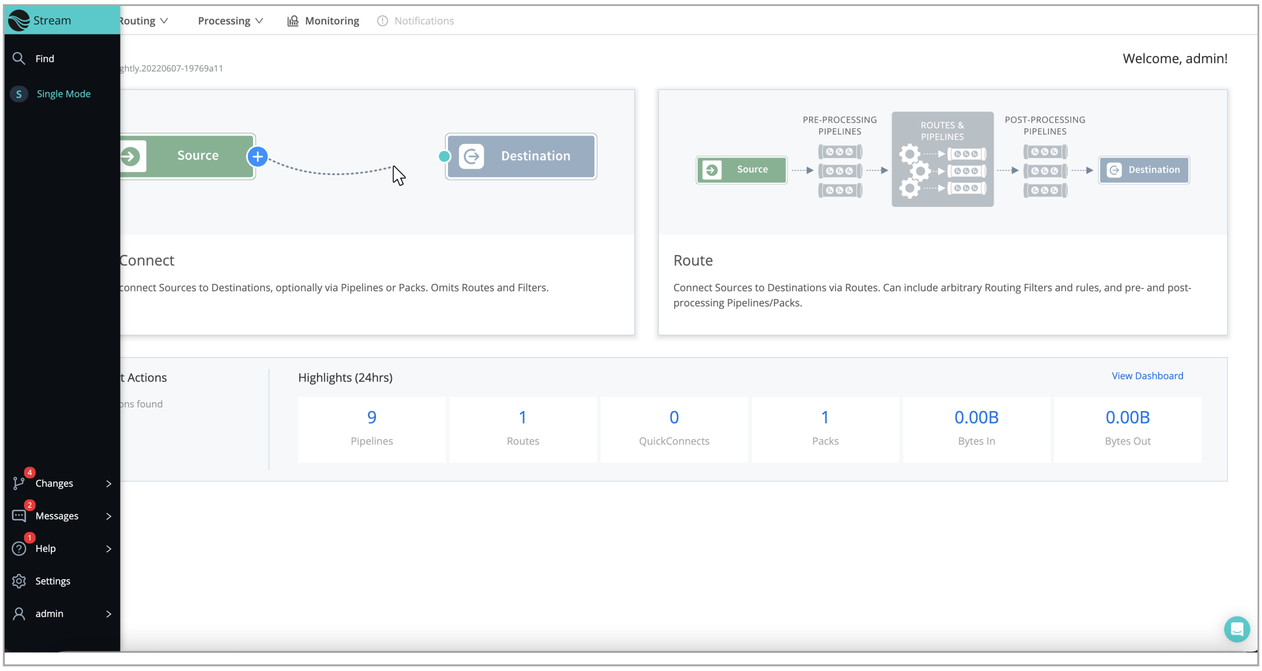
Route

Connect Sources to Destinations via Routes. Can include arbitrary Routing Filters and rules, and pre- and post-processing Pipelines/Packs.

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 consolidated top-menu structure:



Connect

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

Route

Connect Sources to Destinations via Routes. Can include arbitrary Routing Filters and rules, and pre- and post-processing Pipelines/Packs.

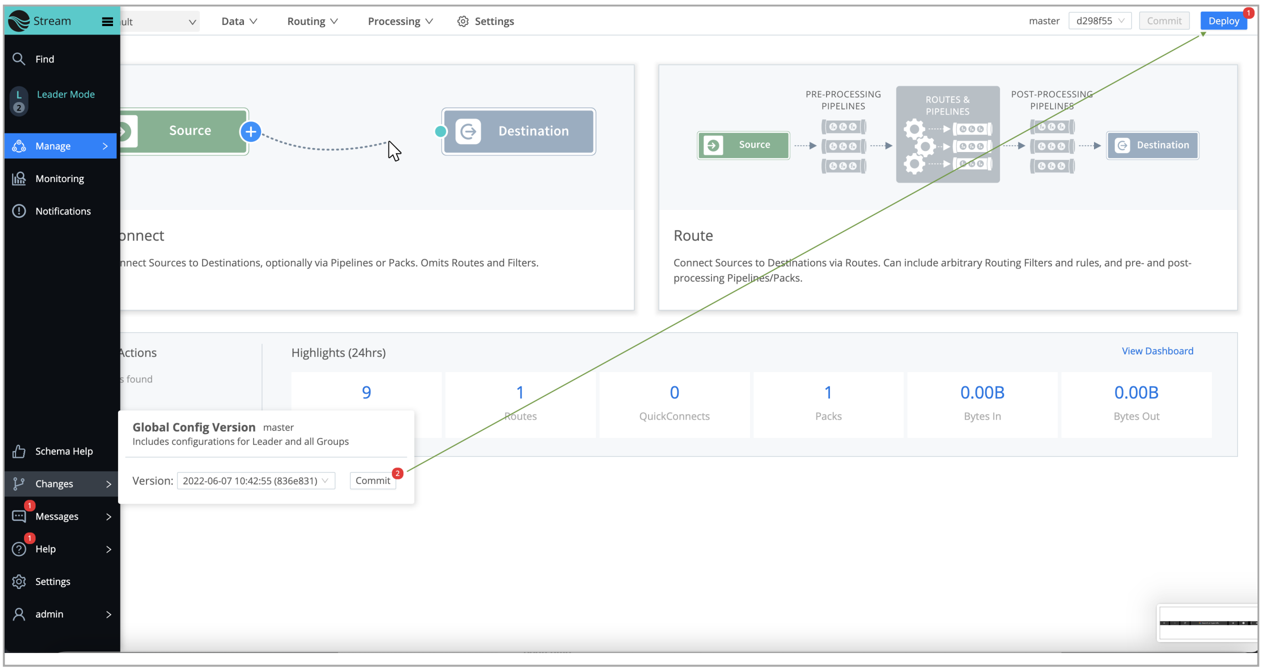
Single-instance deployment: anchored top menu

Commit and Deploy Controls

Distributed mode adds a second set of **Commit** and **Deploy** buttons at the upper right. The division of labor between these version-control buttons versus the global **Commit** button (in the left nav's flyout) is:

- The upper-right buttons commit and deploy configurations for the currently selected Worker Group/Fleet.
- The left 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.



nnect Sources to Destinations, optionally via Pipelines or Packs. Omits Routes and Filters.

Connect Sources to Destinations via Routes. Can include arbitrary Routing Filters and rules, and pre- and post-processing Pipelines/Packs.

Global Config Version master
Includes configurations for Leader and all Groups

Version: 2022-06-07 10:42:55 (836e831)

Commit and Deploy controls – global at left, Group/Fleet at right

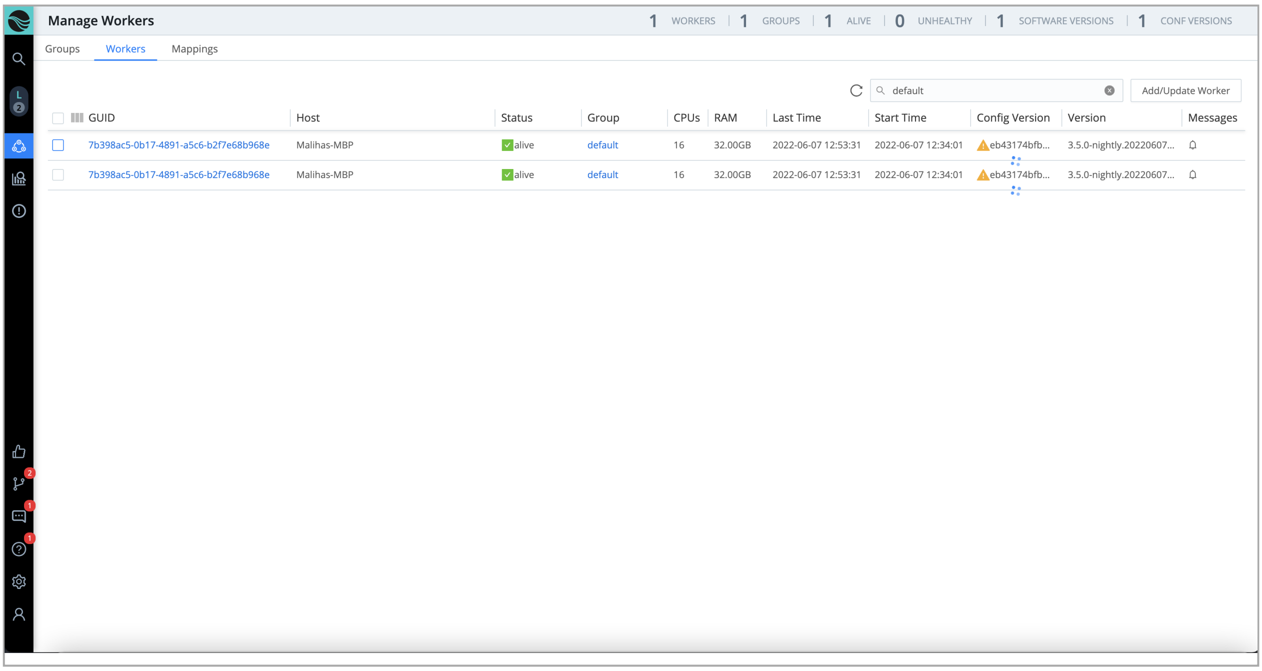
> Distributed mode's repositioning of navigation/menu links also applies to several instructions and screenshots that you'll see throughout this documentation.
>
> Where procedures are written around a single-instance scenario, just click into your appropriate Group to access the corresponding navigation links.

Managing Worker Nodes

Click the left nav's **Manage** tab to open the **Manage Groups** page with three upper tabs: **Groups**, **Workers**, and **Mappings**.

Workers Tab

The **Workers** tab provides status information for each Worker Node in the selected Worker Group.

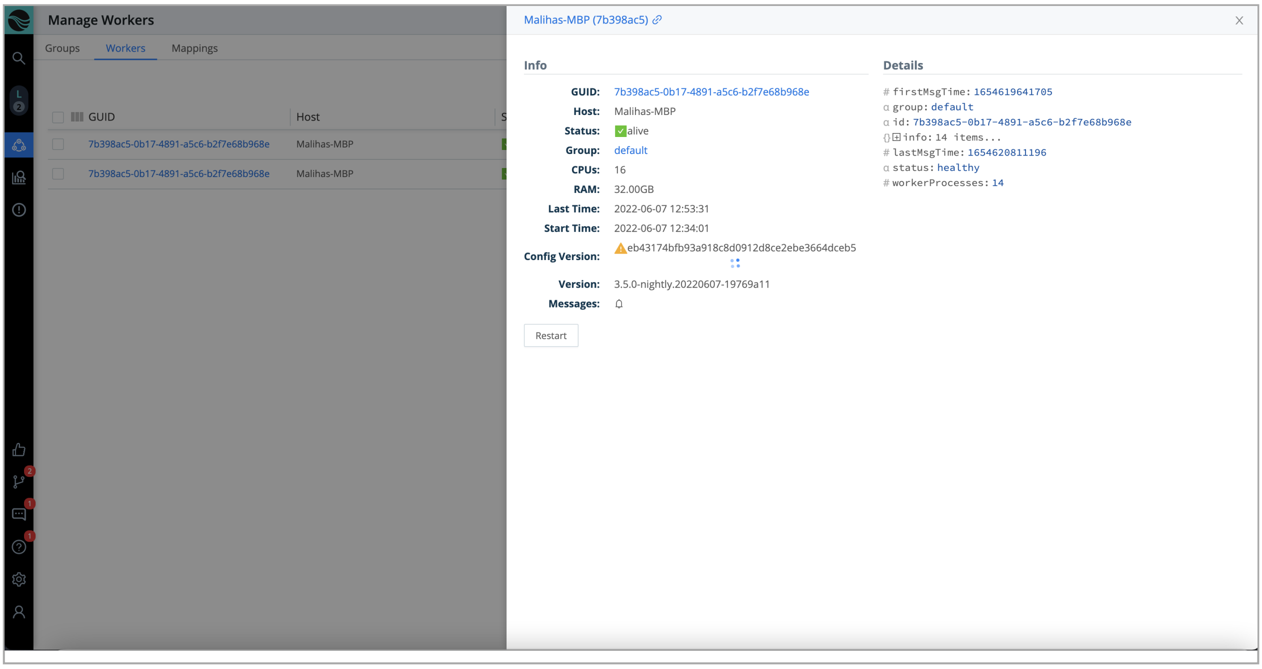


Manage Workers
1 WORKERS | 1 GROUPS | 1 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS
Groups
Workers
Mappings
default
Add/Update Worker

GUID	Host	Status	Group	CPUs	RAM	Last Time	Start Time	Config Version	Version	Messages
7b398ac5-0b17-4891-a5c6-b2f7e68b968e	Malihas-MBP	✓alive	default	16	32.00GB	2022-06-07 12:53:31	2022-06-07 12:34:01	⚠eb43174bfb...	3.5.0-nightly.20220607...	🔔
7b398ac5-0b17-4891-a5c6-b2f7e68b968e	Malihas-MBP	✓alive	default	16	32.00GB	2022-06-07 12:53:31	2022-06-07 12:34:01	⚠eb43174bfb...	3.5.0-nightly.20220607...	🔔

Worker Nodes status/controls

Click anywhere on the row to reveal more details:



Info

GUID: 7b398ac5-0b17-4891-a5c6-b2f7e68b968e
Host: Malihas-MBP
Status: ✓alive
Group: default
CPUs: 16
RAM: 32.00GB
Last Time: 2022-06-07 12:53:31
Start Time: 2022-06-07 12:34:01
Config Version: ⚠eb43174bfb93a918c8d0912d8ce2ebe3664dceb5
Version: 3.5.0-nightly.20220607-19769a11
Messages: 🔔

Details

```
# firstMsgTime: 1654619641705
α group: default
α id: 7b398ac5-0b17-4891-a5c6-b2f7e68b968e
{}⊞ info: 14 items...
# lastMsgTime: 1654620811196
α status: healthy
# workerProcesses: 14
```

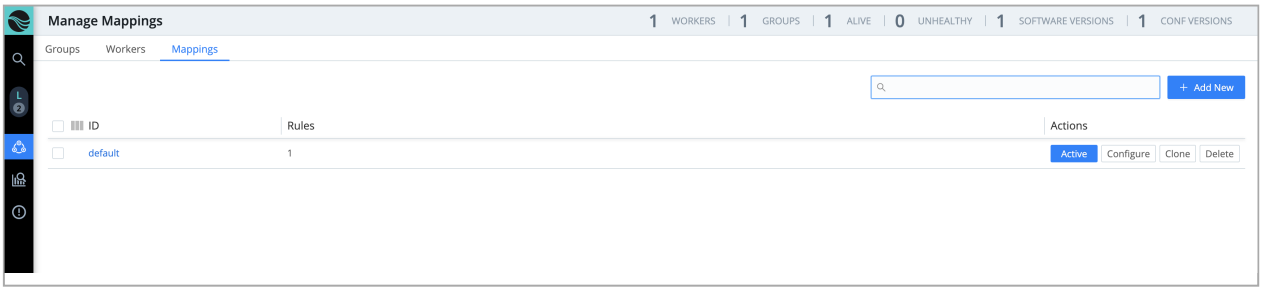
Worker Node Details

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.

Mappings Tab

Click the **Mappings** tab to display status and controls for the active Mapping Ruleset:



Manage Mappings

1 WORKERS | 1 GROUPS | 1 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS

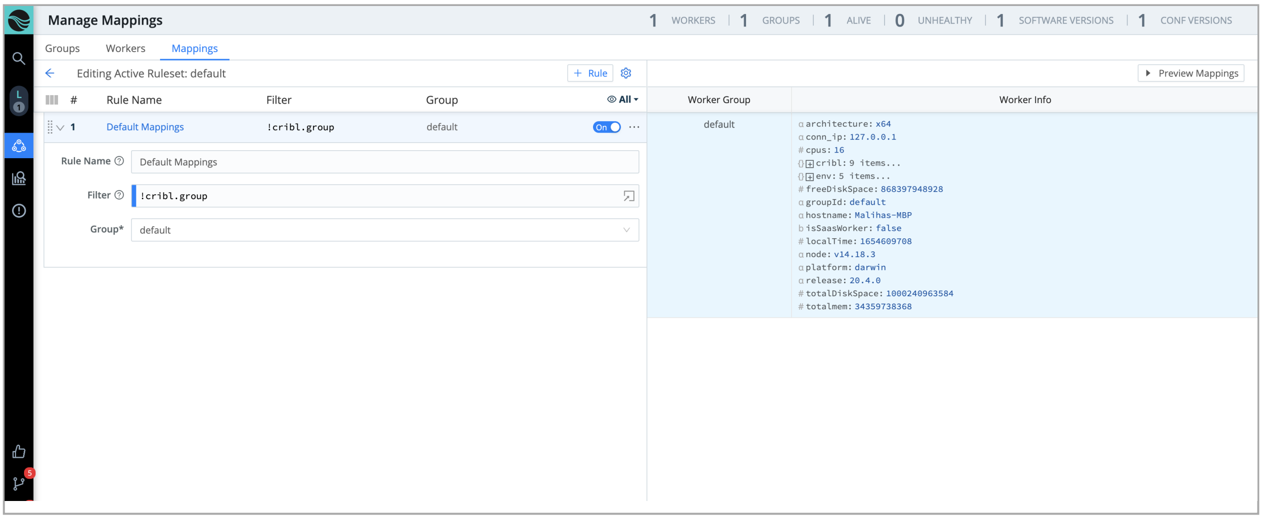
Groups Workers Mappings

+ Add New

ID	Rules	Actions
default	1	Active Configure Clone Delete

Mappings status/controls

Click into a Ruleset to manage and preview its contained Rules:



Manage Mappings

1 WORKERS | 1 GROUPS | 1 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS

Groups Workers Mappings

Editing Active Ruleset: default

+ Rule

#	Rule Name	Filter	Group	All
1	Default Mappings	!cribl.group	default	On

Rule Name: Default Mappings

Filter: !cribl.group

Group*: default

Preview Mappings

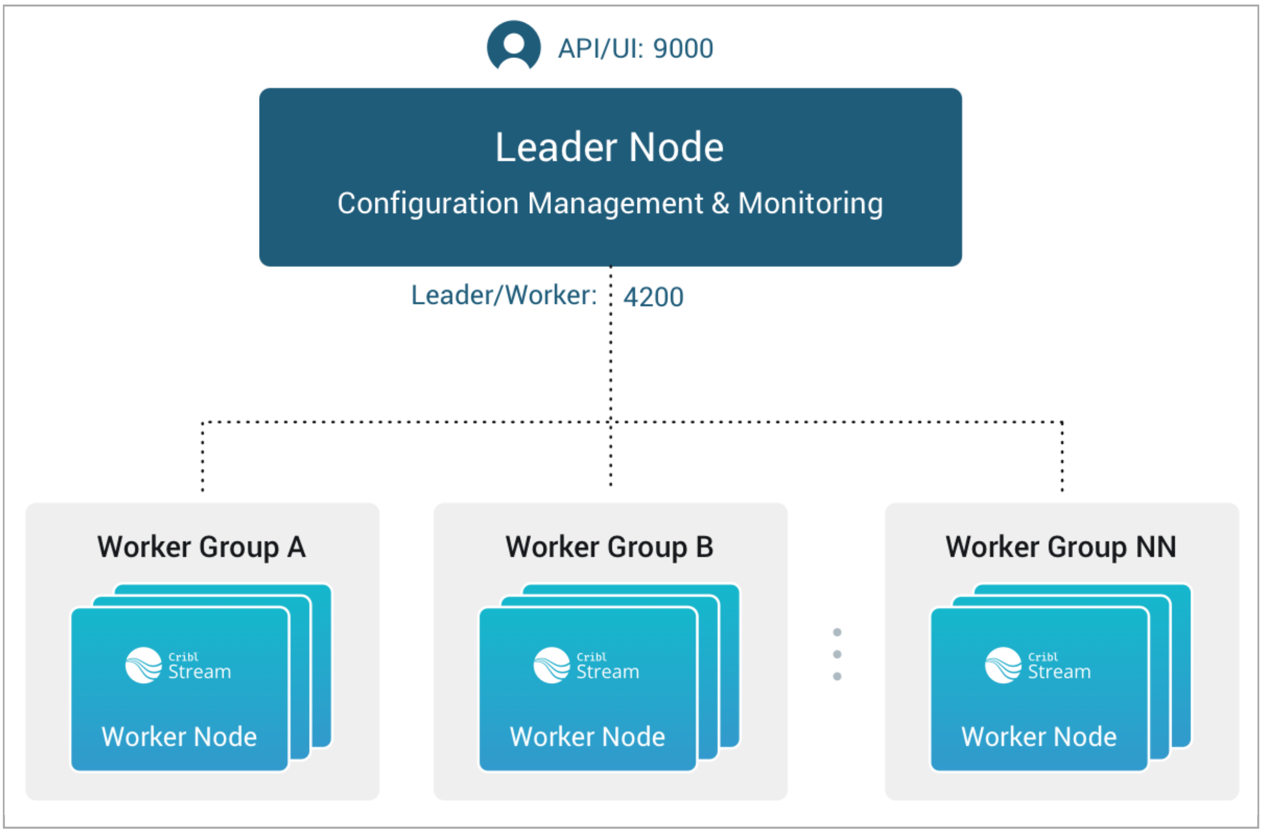
Worker Group	Worker Info
default	architecture: x64 conn_ip: 127.0.0.1 cpus: 16 cribl: 9 items... env: 5 items... freeDiskSpace: 868397948928 groupId: default hostname: Malihas-MBP isSaasWorker: false localTime: 1654609708 node: v14.18.3 platform: darwin release: 20.4.0 totalDiskSpace: 1000240963584 totalmem: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

Managing Ruleset Page

How Do 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 (Heartbeat/Metrics/other).

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.

Worker Groups

Worker Groups facilitate authoring and management of configuration settings for a particular set of Workers. To create a new Worker Group, click **Groups** from the left nav and, from the resulting **Manage Groups** page, click **+ Add New**.

> Configuring multiple Worker Groups, or configuring more than 10 Worker Processes, requires a Cribl Stream Enterprise or Standard license.

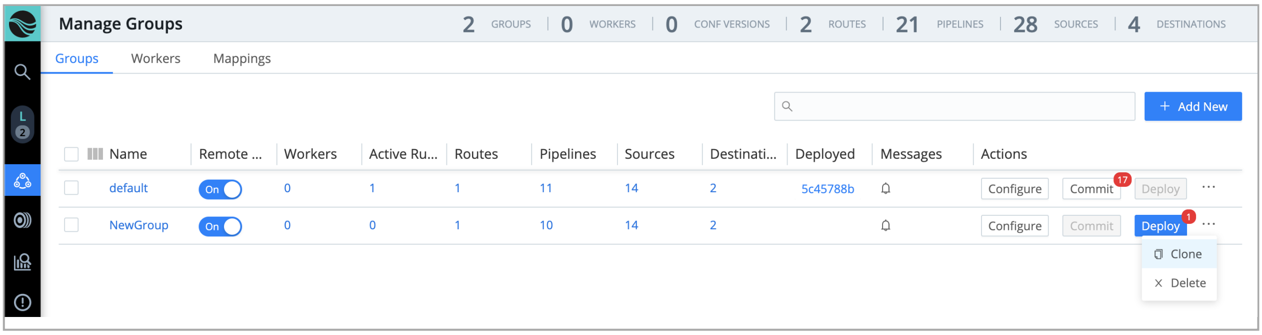
Configuring a Worker Group

Click on newly created Group's **Configure** button to display an interface for **authoring and validating** its configuration. 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 **Manage Groups** 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.



Manage Groups

2 GROUPS | 0 WORKERS | 0 CONF VERSIONS | 2 ROUTES | 21 PIPELINES | 28 SOURCES | 4 DESTINATIONS

□ Name	Remote ...	Workers	Active Ru...	Routes	Pipelines	Sources	Destinati...	Deployed	Messages	Actions
□ default	On	0	1	1	11	14	2	5c45788b		Configure Commit 17 Deploy ···
□ NewGroup	On	0	0	1	10	14	2			Configure Commit Deploy 1 ···

Manage Groups page: controls

> ### Can't Log into the Worker Node as Admin User?
>
> To explicitly set passwords for Worker Groups, see User Authentication.

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 left nav's **Workers** tab to open the **Manage Worker Nodes** page, and then click the **Mappings** upper tab. Click **+ Add New**, give the resulting **New Ruleset** a unique **ID**, and click **Save**.

> The **Mappings** left-nav link appears only when you have started Cribl Stream with global ⚙ **Settings** (lower left) > **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 **+ 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`.

- `CRIBL_DIST_MODE` – `worker | master`. Defaults to `worker` **iff** `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.

  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.

- `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":
["license-free-3.4.0-XYZ98765"]}
```

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.

;

3.3.4. Splunk App Deployment

Getting started with Cribl App for Splunk

> **Cribl App for Splunk for HFs Is Deprecated as of Cribl LogStream v.2.1**
>
> Cribl will continue to support this package, but **customers are advised to begin planning now for the eventual removal of support**.
>
> See Single-Instance Deployment and Distributed Deployment for alternatives.

Deploying Cribl App for Splunk

In a Splunk environment, you can install and configure Cribl Stream as a Splunk app (Cribl App for Splunk). Depending on your requirements and architecture, it can run either on a Search Head or on a Heavy Forwarder. You can use Cribl App for Splunk **cannot** in a Cribl Stream distributed deployment as a Leader, or as a managed Worker.

Regardless of where you run Cribl App for Splunk, if you want to send data from Cribl Stream to a set of Splunk indexers: In the Cribl Stream UI, select **Data** > **Destinations** > Splunk Load Balanced, then enter the required information.

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.

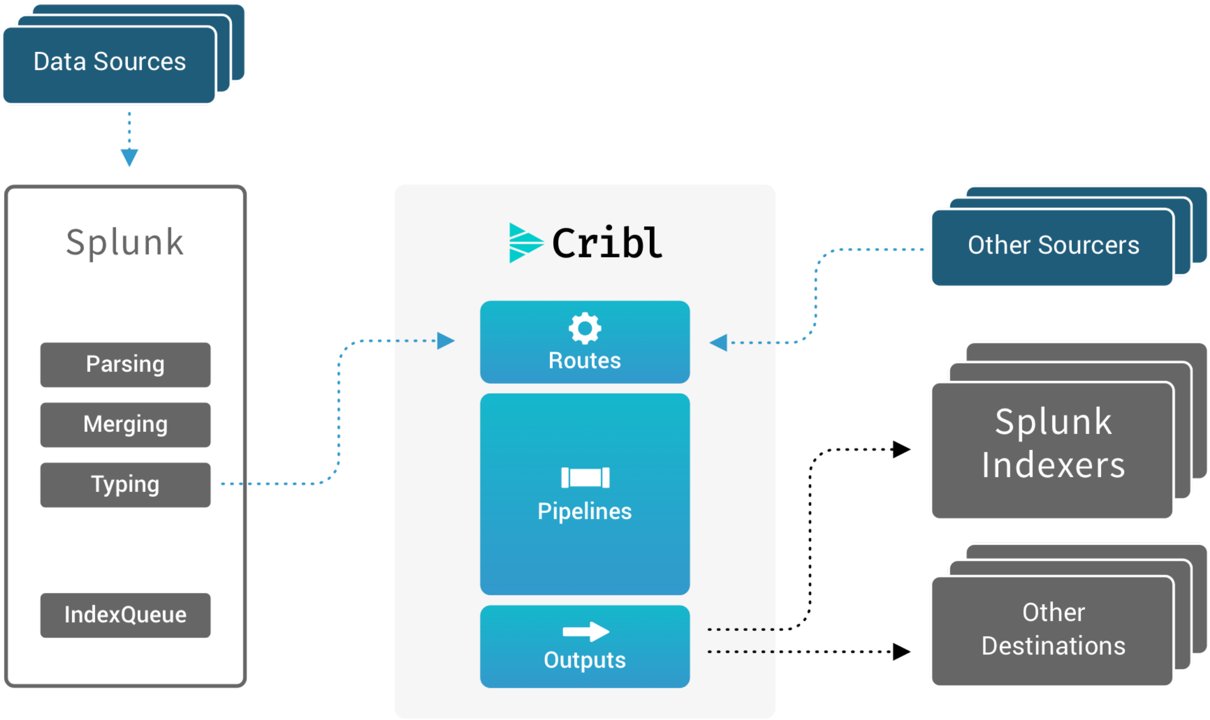
- Ensure that ports `10000`, `10420`, and `9000` are available. See the Requirements section for more info.
- Get the bits here, and install as a regular Splunk app.
- Restart the Splunk instance.
- Go to `https://<instance>/en-US/app/cribl` or `https://<instance>:9000`, and log in with Splunk **admin** role credentials.

Typical Use Cases for Search Head Mode

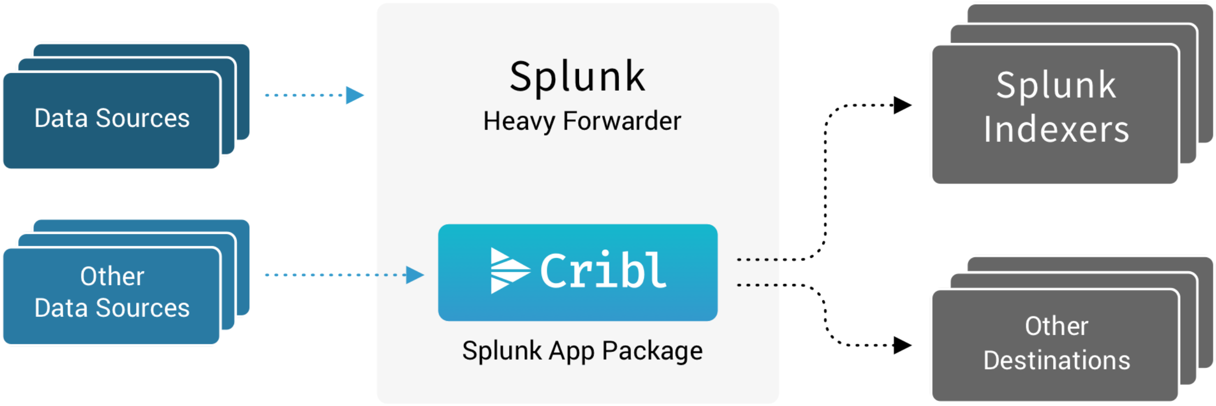
- Working with search results in a Cribl Stream pipeline.
- Sending search results to any Destination supported by Cribl Stream.

Running on a Heavy Forwarder (HF)

When running on an HF, Cribl Stream is set to **mode-hwf**. It receives events from the local Splunk process per routing configurations in `props.conf` and `transforms.conf`. Data is parsed and processed first by Splunk pipelines, and then by Cribl Stream. By default, all data except internal indexes is routed out right after the Typing pipeline.



Cribl Stream is capable of accepting data **streams** (unbroken events) or **events** from other sources. In this case, the HF will deliver **events** locally to Cribl Stream, which processes them and sends them to one or more destinations downstream. When receivers are Splunk indexers, Cribl Stream can also load-balance across them.



Installing the Cribl App for Splunk on an HF

- Select an instance on which to install.
- Ensure that ports `10000`, `10420`, and `9000` are available. See here.
- Get the bits here, and install as a regular Splunk app.
- Set Cribl to **mode-hwf**: `$SPLUNK_HOME/etc/apps/cribl/bin/cribl mode-hwf`.

> The `SPLUNK_HOME` environment variable must be defined.

- Restart the Splunk instance.
- Go to `https://<instance>:9000` and log in with Splunk **admin** role credentials.

> ### Note About Splunk Warnings
>
> If you come across messages similar to the following example, on startup or in logs, please ignore them. They are benign warnings.
>
> ```
> Invalid value in stanza [route2criblQueue]/[hecCriblQueue] in
> /opt/splunk/etc/apps/cribl/default/transforms.conf, line 11: (key:
> DEST_KEY, value: criblQueue) / line 24: (key: DEST_KEY, value: $1)
> ```

Relevant configurations in Cribl App for Splunk on an HF

When Cribl App for Splunk is installed on an HF (in `mode-hwf`), below are the **relevant sections** in configuration files that enable Splunk to send data to Cribl Stream:

apps/cribl/default/outputs.conf

```
[tcpout]
disabled = false
defaultGroup = cribl

[tcpout:cribl]
server=127.0.0.1:10000
sendCookedData=true
useACK = false
negotiateNewProtocol = false
negotiateProtocolLevel = 0
```

apps/cribl/default/inputs.conf

```
[splunktcp]
route=has_key:_replicationBucketUUID:replicationQueue;has_key:_dstrx:typingQueue;has_ke
```

apps/cribl/default/transforms.conf

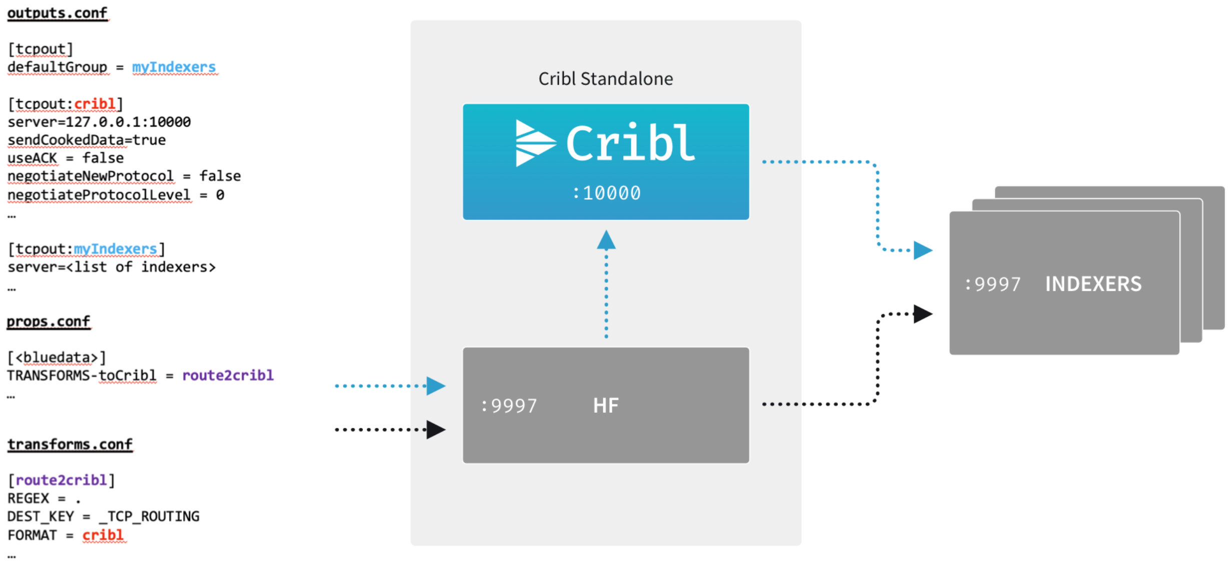
```
[route2cribl]
SOURCE_KEY = _MetaData:Index
REGEX = ^[^_]
DEST_KEY = _TCP_ROUTING
FORMAT = cribl

[route2criblQueue]
SOURCE_KEY = _MetaData:Index
REGEX = ^[^_]
DEST_KEY = queue`
FORMAT = criblQueue
```

apps/cribl/default/props.conf

```
[default]
TRANSFORMS-cribl = route2criblQueue, route2cribl
```

Configuring Cribl Stream with a Subset of Your Data

The `props.conf` stanza above will apply its transforms to **everything**. Depending on your requirements, you might want to target only a subset of your sources, sourcetypes, or hosts. For example, the diagram below shows the **effective** configurations of `outputs.conf`, `props.conf`, and `transforms.conf` to send `<bluedata>` events through Cribl Stream.



```
outputs.conf

[tcpout]
defaultGroup = myIndexers

[tcpout:cribl]
server=127.0.0.1:10000
sendCookedData=true
useACK = false
negotiateNewProtocol = false
negotiateProtocolLevel = 0
…

[tcpout:myIndexers]
server=<list of indexers>
…

props.conf

[<bluedata>]
TRANSFORMS-toCribl = route2cribl
…


transforms.conf

[route2cribl]
REGEX = .
DEST_KEY = _TCP_ROUTING
FORMAT = cribl
…
```

;

3.3.5. 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 fleet of any number of Nodes 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.

Requirements

All Worker Nodes' hosts must keep port 4200 open for ongoing management by the Leader. 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/Bootstrap New Worker

1. From Cribl Stream's left nav, select **Workers**.

2. On the resulting **Manage Worker Nodes** page, click **+ Add/Update Worker** at the upper right.

3. Select **Bootstrap new** from the menu, as shown in the composite screenshot below.

4. In the resulting **Add Worker** modal, the **Cribl Stream package location** defaults to `Cribl CDN`. If desired, change this to `Download URL`. (For details about this option, see Adding Download URL.)

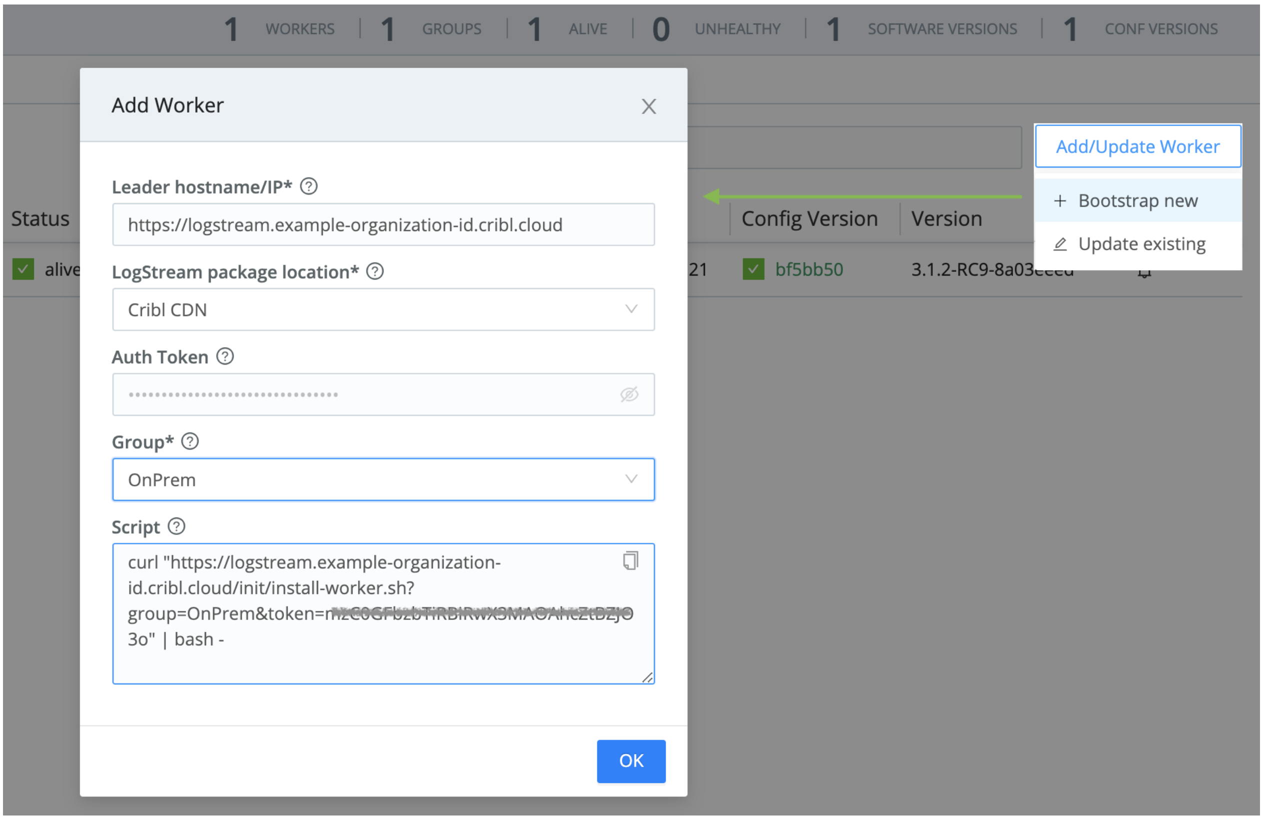
5. As needed, correct the target **Group**, as well as the **Leader hostname/IP** (URI).

6. Copy the resulting script to your clipboard.

7. Click either **OK** or **Cancel** to close the modal.

8. 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.



Leader hostname/IP*: https://logstream.example-organization-id.cribl.cloud

LogStream package location*: Cribl CDN

Auth Token: ••••••••••••••••••••••••••••••••

Group*: OnPrem

Script:

```
curl "https://logstream.example-organization-
id.cribl.cloud/init/install-worker.sh?
group=OnPrem&token=[redacted]
3o" | bash -
```

Add/Bootstrap Worker UI option (composite)

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. Follow steps 1–2 in Add/Bootstrap New Worker above.

2. From the **+ Add/Update Worker** menu, select **Update existing**.

3. In the resulting **Update Worker** modal, select the new target **Group** for this Worker.

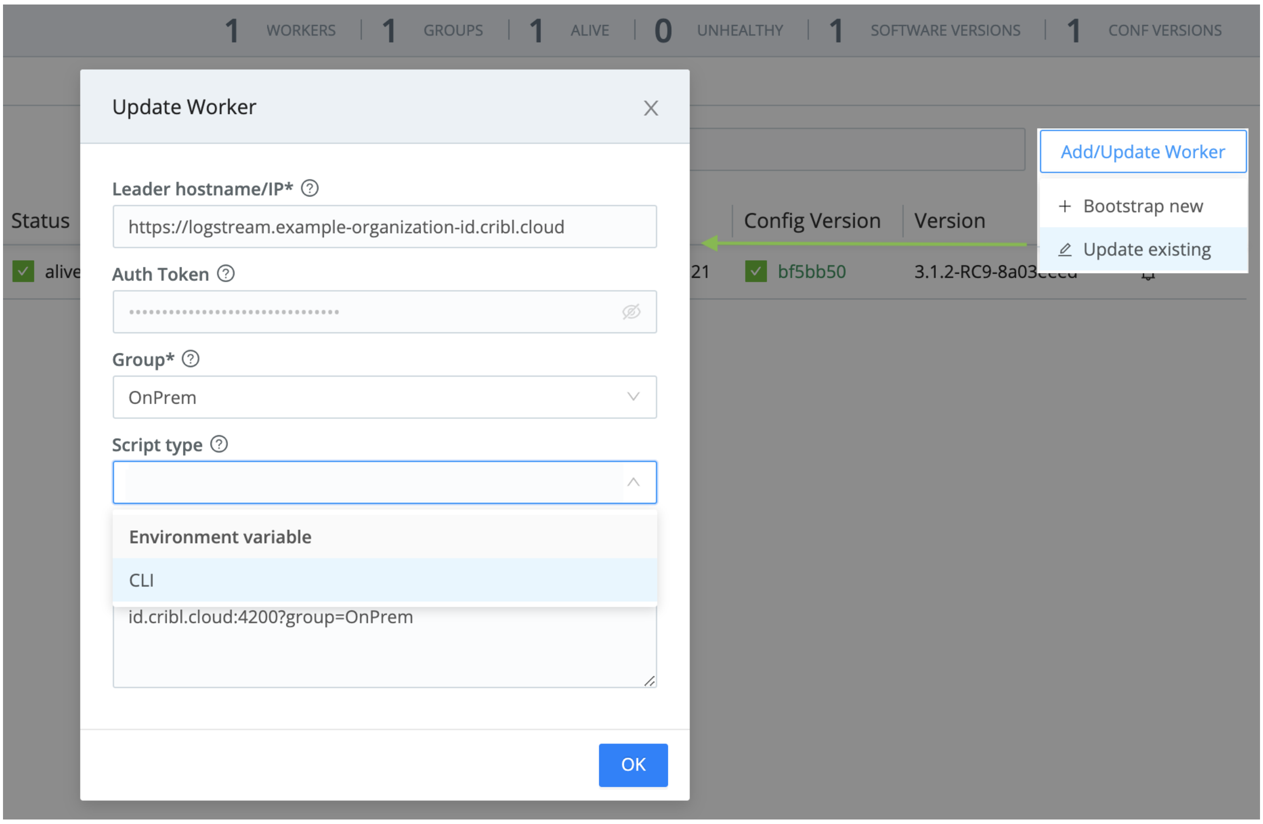
4. As needed, correct or change the **Leader hostname/IP** and/or **Leader port number**.

5. 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`.

6. Copy the resulting script to your clipboard.

7. Click either **OK** or **Cancel** to close the modal.

8. 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.



Update Worker

Leader hostname/IP*: https://logstream.example-organization-id.cribl.cloud

Auth Token: ••••••••••••••••••••••••••••••••

Group*: OnPrem

Script type:

id.cribl.cloud:4200?group=OnPrem

Update Worker UI option (composite)

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 `criblmaster` . You can find this secret in the Leader Node's **Distributed Settings** section.

STRING	REQUIRED?	DESCRIPTION
`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` .

Example HTTP Request

```
GET http://<leader hostname or IP>:9000/init/install-worker.sh?token=79364d6e-dead-beef-4c6e-554445664867
```

> 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 ${CRIBL_VERSION} | cut -d '-'
-f 1)/${FILE}"
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-password
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-dead-beef-4c6e-554445664867
```

> 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-dead-beef-4c6e-554445664867 | bash -
```

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-
dead-beef-4c6e-544445664867&download_url=https://<your_internal_webserver>/cribl-
2.2.0-4589617e-linux-x64.tgz" | sh -
```

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=default&token=JOINTOKEN&download_url=/root/&user=cribl&install_dir=/opt/cribl
   | bash -
   ```

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=criblmaster" | bash -
```

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=criblmaster
```

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 this feature:

- 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 this feature 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-
token> | sh -
```

;

3.3.6. 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.

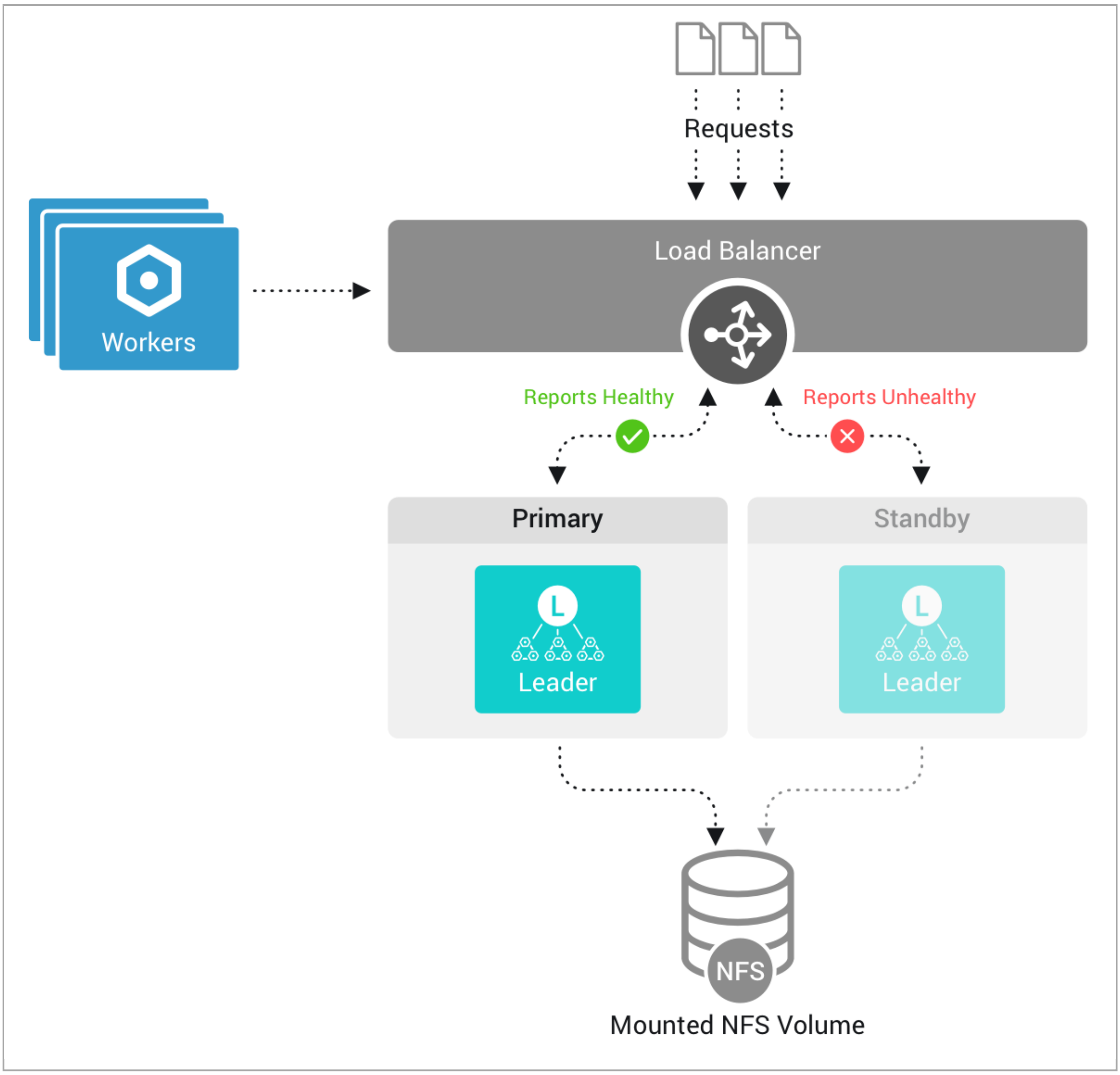
> Configuring a backup Leader requires a Cribl Stream Enterprise or Standard license.

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 same the 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 following configuration:

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 global ⚙ **Settings** (lower left) > **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.

Load Balancers

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

The following load balancers support health checks:

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

Recommended Configuration

Use the latest NFS client across all Leaders. If you are on AWS, we recommend using 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.

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.

Configuring a Second Leader Node

You can configure your second Leader Node in the following ways. These configuration options are similar to configuring the primary Leader Node:

- Using the UI
- Updating the YAML config file
- Using the Command Line
- Using Environment Variables

Using the UI

1. In global ⚙ **Settings** (lower left) > **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 location of the NFS directory to support Leader failover (e.g., `(/mnt/cribl)`).

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.

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 a second 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

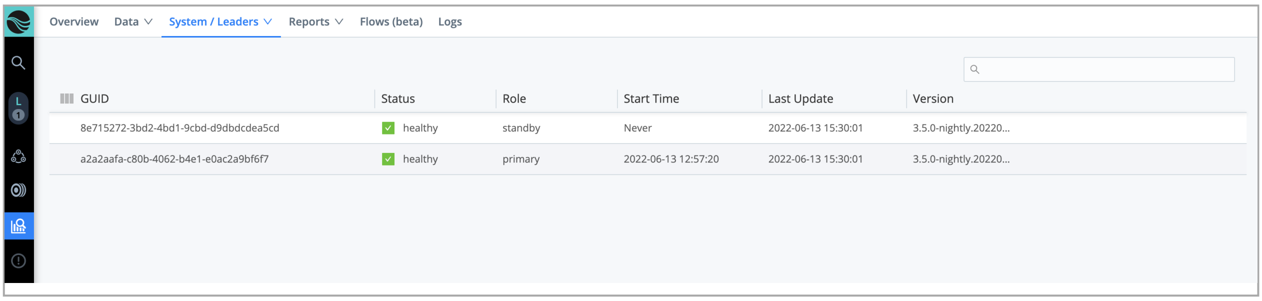
You can configure a second Leader Node via the following environment variables:

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

For further variables, see Environment Variables.

Monitoring the Leader Nodes

To view the status of your Leader Nodes, select **Monitoring** (side or top nav) > **System** > **Leaders**.



GUID	Status	Role	Start Time	Last Update	Version
8e715272-3bd2-4bd1-9cbd-d9dbdcdea5cd	✓ healthy	standby	Never	2022-06-13 15:30:01	3.5.0-nightly.20220...
a2a2aafa-c80b-4062-b4e1-e0ac2a9bf6f7	✓ healthy	primary	2022-06-13 12:57:20	2022-06-13 15:30:01	3.5.0-nightly.20220...

Monitoring Leader Nodes

Upgrading

When upgrading:

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

;

3.3.7. 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 **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.

;

3.4. Orchestrated Deployment

3.4.1. Kubernetes/Helm Deployment

Cribl's leader and workergroup Helm charts provide a fast way to deploy a distributed Cribl Stream environment to a Kubernetes cluster.

Prerequisites

Helm version 3 is required to use these charts.

To install Helm on (e.g.) a Mac, using Homebrew:

```
brew install helm
```

Find instructions for other operation systems in Helm's installation documentation.

Deploying

If you haven't done so already, create a namespace. Our documentation example uses `logstream`.

```
kubectl create namespace logstream
```

Add the Cribl Helm repo.

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

The following example creates a distributed deployment with two autoscaled Worker Groups/Fleets, `pcilogs` and `system-metrics`. It uses an auth token of `ABCDEF01-1234-5678-ABCD-ABCDEF012345`, sets an admin password, and installs our license:

```
helm install ls-leader cribl/logstream-leader \
  --set "config.groups={pcilogs,system-metrics}" \
  --set config.token="ABCDEF01-1234-5678-ABCD-ABCDEF012345" \
  --set config.adminPassword="<admin password>" \
  --set config.license="<license key>" \
  -n logstream

helm install ls-wg-pci cribl/logstream-workergroup \
  --set config.host="ls-leader-internal" \
  --set config.tag="pcilogs" \
  --set config.token="ABCDEF01-1234-5678-ABCD-ABCDEF012345" \
  -n logstream

helm install ls-wg-system-metrics cribl/logstream-workergroup \
  --set config.host="ls-leader-internal" \
  --set config.tag="system-metrics" \
  --set config.token="ABCDEF01-1234-5678-ABCD-ABCDEF012345" \
  -n logstream
```

Running Distributed on a Free License

If you do not specify a license in your install with `config.license`, and you want to run distributed, you'll need to go to Cribl Stream's **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 > Distributed** page to select **Mode:** `Leader`.)

Upgrading

Upgrading Cribl Stream to new bits via Helm is easy. Sync up your repo to the origin, and then upgrade each chart version. The example below updates to the current version, but you can append `--version X.Y.Z` if you want to specify a particular version.

```
helm repo update
helm upgrade ls-leader cribl/logstream-leader -n logstream
helm upgrade ls-wg-pci cribl/logstream-workergroup -n logstream
helm upgrade ls-wg-system-metrics cribl/logstream-workergroup -n logstream
```

;

3.4.2. K8s Leader Deployment

Boot a fully provisioned Leader Node via Helm

This page outlines how to deploy a Cribl Stream Leader Node (or single instance) to AWS via Kubernetes, using a Cribl-provided 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

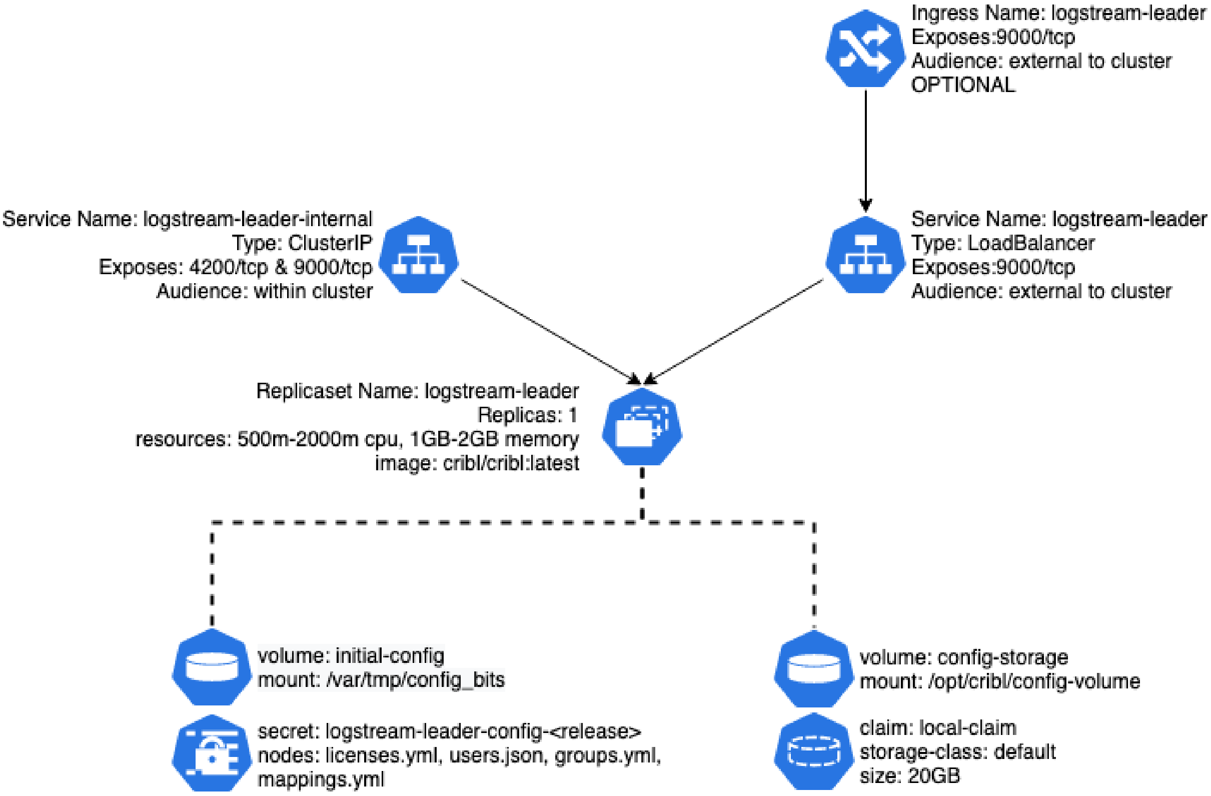
This chart replaces the logstream-master chart, which was deprecated as of v.2.9.9. See Migration below for instructions on migrating to this new chart to access features newly provided in this and future versions.

New Capabilities

- Supports Cribl Stream v.3.5.2 (default version).
- Supports the `nodeSelector` configuration option for managing pod scheduling.
- Supports using a fixed IP address for `LoadBalancer`s in both created services, via the `service.internalLoadBalancerIP` and `service.externalLoadBalancerIP` options. (A fixed IP address is not universally supported across K8s implementations; check your implementation before configuring this option.)

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:

```
[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:

```
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 below).

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

Values to Override

This section covers the most likely values to override. To see the full scope of values available, run: `helm show values cribl/logstream-leader`

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`config.adminPassword`	String	[No default]	The password you want to assign to the admin user.
`config.token`	String	[No default]	The auth key you want to set up for Worker access. If you set this value, the Cribl Stream instance will be configured only as a Leader server for a distributed deployment. (You can also configure this later via the Cribl Stream UI, after launching the instance in single-instance mode.)
`config.license`	String	[No default]	The license for your Cribl Stream instance. If you do not set this, it will default to the Free license. You can change this in the Cribl Stream UI as well.
`config.groups`	List	[No default]	Array of Worker Group names to configure for the Leader instance. This will create a mapping for each Group, which looks for the tag `<groupname>`, and will create the basic structure of each Group's configuration.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`config.scName`	String	\<default StorageClass name>	The StorageClass name for all of the persistent volumes.
`config.rejectSelfSignedCerts`	Number	`0`	Either `0` (allow self-signed certificates) or `1` (deny self-signed certs).
`config.healthPort`	Number	`9000`	The port to use for health checks (readiness/live).
`config.healthScheme`	String	`HTTP`	The scheme to use for health checks. Supports `HTTP` or `HTTPS`.
`service.internalType`	ClusterIP	[No default]	The type to use for the `<release>-leader-internal` service. In 2.4.5+, this is set to `ClusterIP` by default. If you have any Worker Groups/Fleets outside of the Kubernetes cluster where the Leader lives, you'll need to change this to `NodePort` or `LoadBalancer` to expose it outside of the cluster.
`service.internalLoadBalancerIP`	IP address	[No default]	If the `service.internalType` is set to `LoadBalancer`, specifies the IP address to use for the load-balancer service interface. Before configuring, check whether your K8s implementation supports fixed IP addresses.
`service.externalType`	Load Balancer	[No default]	The type to use for the user-facing `<release>-leader` service. If `ingress.enable` is set, this will be force-set to `NodePort`, to work with the ingress.
`service.externalLoadBalancerIP`	IP address	[No default]	If the `service.externalType` is set to `LoadBalancer`, specifies the IP address to use for the load-balancer service interface. Before configuring, check whether your K8s implementation supports fixed IP addresses.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION

`service.ports`	Array of Maps	`- name: api`
		`  port: 9000`
		`  protocol: TCP`
		`  external: true`
		`- name: leadercomm`
		`  port: 4200`
		`  protocol: TCP`
		`  external: false`
			The ports to make available, both in the deployment and in the service. Each "map" in this list needs the following values set:
			name
			A descriptive name, identifying what the port is being used for.
			port
			The container port to be made available.
			protocol
			The protocol in use for this port (UDP or TCP).
			external
			Set to `true` to `expose` the port on the external service, or `false` to not expose it.

`service.annotations`	Object	[No default]	Annotations for the service component – this is where you'll want to put load-balancer–specific configuration directives.
`criblImage.tag`	String	`3.5.2`	The container image tag to pull from. Cribl will increment this tag per Cribl Stream version. By default, this will use a version equivalent to the chart's `appVersion` value. You can override this with `latest` to get the latest Cribl Stream version, or with a specific Cribl Stream version number (like "3.5.1").
`consolidate_volumes`	boolean	[No default]	If this value exists, and the `helm` command is `upgrade`, this will use the split volumes that we created in charts before 2.4 and consolidate them down to one config volume. This is a **one-time** event.
`nodeSelector`	Object	[No default]	Add `nodeSelector` values to define the nodes on which pods are scheduled. For details and allowed values, see K8s' Assigning Pods to Nodes topic.

Extra Configuration Options

The links here point to configuration details on our GitHub repo.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
extraVolumeMounts	Object	[No default]	Additional volumes to mount in the container.
extraSecretMounts	Array	[No default]	Pre-existing Secrets to mount within the container.
extraConfigmapMounts	Object	[No default]	Pre-existing ConfigMaps to mount within the container.
extraContainers	Object	[No default]	Additional containers to run alongside the main Cribl Stream container. This allows you to implement the standard Sidecar pattern with the Cribl Stream Helm charts.
extraInitContainers	Object	[No default]	Additional containers to run ahead of the primary container in the pod.
securityContext.runAsUser	Number	0	User ID to run the container processes under.
securityContext.runAsGroup	Number	0	Group ID to run the container processes under.
envValueFrom	Object	[No default]	Environment variables to be exposed from the Downward API.
env	Array	[No default]	Additional static environment variables.
ingress.enable	boolean	false	Enable Ingress in front of the external service. Setting this to `true` changes the external service to type `NodePort`, and creates an ingress that connects to it.
ingress.annotations	Object	[No default]	If `ingress.enable` is set to `true`, this is where annotations to configure the specific ingress controller. (NOTE: Ingress is supported only on Kubernetes 1.19 and later clusters).

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:

```
    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: "<bucket
name>"
    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.

> More options are coming here!

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='lebs-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 global ⚙ **Settings** (lower left) > **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 global ⚙ **Settings** (lower left) > **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; tar
cf - ." > cribl_backup.tar
```

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 "cd
/opt/cribl/config-volume/; tar xf -"
```

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-volume"
```

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: criblmaster
  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-leader
```

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
"CRIBL_.*START"
CRIBL_BEFORE_START_CMD_1:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/licenses.yml
]; then mkdir -p $CRIBL_VOLUME_DIR/local/cribl ; cp
/var/tmp/config_bits/licenses.yml $CRIBL_VOLUME_DIR/local/cribl/licenses.yml; fi
CRIBL_BEFORE_START_CMD_2:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/mappings.yml
]; then mkdir -p $CRIBL_VOLUME_DIR/local/cribl;  cp /var/tmp/config_bits/groups.yml
$CRIBL_VOLUME_DIR/local/cribl/groups.yml; cp /var/tmp/config_bits/mappings.yml
$CRIBL_VOLUME_DIR/local/cribl/mappings.yml; fi
CRIBL_AFTER_START_CMD_1:       [ ! -f $CRIBL_VOLUME_DIR/users_imported ] && sleep 20
&& cp /var/tmp/config_bits/users.json $CRIBL_VOLUME_DIR/local/cribl/auth/users.json
&& touch $CRIBL_VOLUME_DIR/users_imported
```

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}/${_time:%Y}/${_time:%m}/
      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/group1/local/cribl/jobs.yml"
```

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 directory
tree exists
  cp /var/tmp/job-config/jobs.yml /opt/cribl/config-volume/groups/group1/local/cribl
# copy the file
  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 [ ! -e
$FLAG_FILE ]; then mkdir -p /opt/cribl/config-volume/groups/group1/local/cribl; cp
/var/tmp/job-config/jobs.yml /opt/cribl/config-volume/groups/group1/local/cribl;
touch $FLAG_FILE; fi"
```

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 `PersistentVolume`s and `PersistentVolumeClaim`s 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.

;

3.4.3. K8s 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.

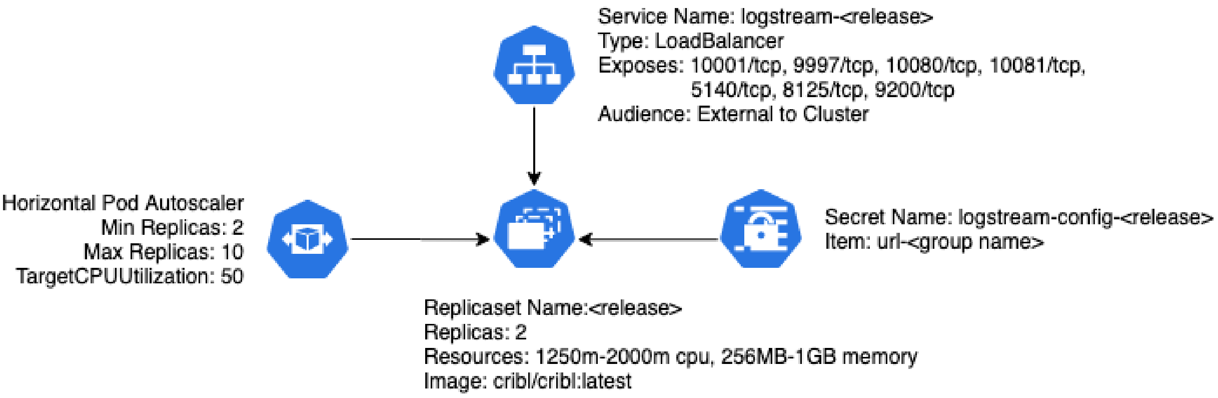
> This chart will deploy only a Cribl Stream Worker Group, whose functioning depends on the presence of a Cribl Stream Leader Node. To deploy the Leader, see Kubernetes Leader Deployment.

New Capabilities

- Supports Cribl Stream v.3.5.2 (default version).
- Supports the `nodeSelector` configuration option for managing pod scheduling.
- Supports use of AWS IAM Roles for Worker Group/Fleet RBAC.
- Supports using a fixed IP address for `LoadBalancer` s in the created service, via the `service.loadBalancerIP` option. (A fixed IP address is not universally supported across K8s implementations; check your implementation before configuring this option.)

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:

```
[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:

```
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/`

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 below).

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

Values to Override

This section covers the most likely values to override. To see the full scope of values available, run: `helm show values cribl/logstream-workergroup.`

> From version 3.0 onward, Cribl Stream's former "master" application components are renamed "leader."

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`config.group`	String	`kubernetes`	Tag/group to include in the URL (included as both a group value and a tag value).
`config.tag`	[Deprecated]	[Deprecated]	The tag/group to include in the URL. (This option is deprecated, but is still supported for backward compatibility.
`config.token`	String	`criblleader`	The authentication token for your Cribl Stream Leader.
`config.host`	String	`logstream-leader-internal`	The resolvable hostname of your Cribl Stream Leader.
`config.rejectSelfSignedCerts`	Number	`0`	One of: `0` – allow self-signed certs, or `1` – deny self-signed certs.
`config.tlsLeader.enable`	Boolean	false	Enable TLS connectivity from the workergroup to its Leader Node.
`config.hostNetwork`	Object	false	Configures the workergroup to use the K8s host network instead of the container network.
`config.probes`	Boolean	true	Enables (true) or disables (false) the liveness and readiness probes.
`service.type`	String	`LoadBalancer`	The type of service to create for the workergroup.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`service.loadBalancerIP`	IP address	[No default]	If `service.type` is set to `LoadBalancer`, specifies the IP address to use for the load-balancer service interface. Before configuring, check whether your K8s implementation supports fixed IP addresses.

`service.ports`	Array of Maps	`- name: tcpjson`
		`port: 10001`
		`protocol: TCP`
		`- name: s2s`
		`port: 9997`
		`protocol: TCP`
		`- name: http`
		`port: 10080`
		`protocol: TCP`
		`- name: https`
		`port: 10081`
		`protocol: TCP`
		`- name: syslog`
		`port: 5140`
		`protocol: TCP`
		`- name: metrics`
		`port: 8125`
		`protocol: TCP`
		`- name: elastic`
		`port: 9200`
		`protocol: TCP`
			The ports to make available, both in the deployment and in the service. Each "map" in this list needs the following values set:
			name
			A descriptive name, identifying what the port is being used for.
			port
			The container port to make available.
			protocol
			The protocol in use for this port (UDP or TCP).

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`service.annotations`	Object	[No default]	Annotations for the service component – this is where you'll want to put load-balancer–specific configuration directives.
`criblImage.tag`	String	`3.5.2`	The container image tag to pull from. Cribl will increment this tag per Cribl Stream version. By default, this will use a version equivalent to the chart's `appVersion` value. You can override this with `latest` to get the latest Cribl Stream version, or with a specific Cribl Stream version number, such as `3.5.1`.
`autoscaling.minReplicas`	Number	`2`	The minimum number of Cribl Stream pods to run.
`autoscaling.maxReplicas`	Number	`10`	The maximum number of Cribl Stream pods to scale up to.
`autoscaling.targetCPUUtilizationPercentage`	Number	`50`	The CPU utilization percentage that triggers scaling action.
`rbac.create`	Boolean	`false`	Enable Service Account, Cluster Role, and Role Binding creation.
`rbac.resources`	List	`["pods"]`	Set the resource boundary for the role being created (K8s resources).
`rbac.verbs`	List	`["get", "list"]`	Set the API verbs allowed the role (default: `read ops`).
`rbac.annotations`	Object	[No default]	Annotations for the RBAC component of an AWS EKS environment where you want its pods to use IAM Roles. First, set `rbac.create` to `true`. Then follow the applicable procedures for EKS and Cribl Stream, including specifying a value for the `rbac.annotations` key.
`rbac.apiGroups`	List	`["Core"]`	Set the apiGroups in roles rules
`nodeSelector`	Object	[No default]	Add `nodeSelector` values to define the nodes on which pods are scheduled. For details and allowed values, see K8s' Assigning Pods to Nodes topic.

Extra Configuration Options

The links here point to configuration details on our GitHub repo.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
extraVolumeMounts	Object	[No default]	Additional volumes to mount in the container.
extraSecretMounts	Array	[No default]	Pre-existing Secrets to mount within the container.
extraConfigmapMounts	Object	[No default]	Pre-existing ConfigMaps to mount within the container.
extraContainers	Object	[No default]	Additional containers to run alongside the main Cribl Stream container. This allows you to implement the standard Sidecar pattern with the Logstream Helm charts.
extraInitContainers	Object	[No default]	Additional containers to run ahead of the primary container in the pod.
securityContext.runAsUser	Number	0	User ID to run the container processes under.
securityContext.runAsGroup	Number	0	Group ID to run the container processes under.
envValueFrom	Object	[No default]	Environment variables to be exposed from the Downward API.
env	Array	[No default]	Additional static environment variables.
deployment	String	`deployment`	One of: `deployment` to deploy as a Deployment Set; or `daemonset` to deploy as a DaemonSet.
rbac.extraRules	Object	[No default]	Additional RBAC rules to put in place.

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.

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 Leadermast `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:

```
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.

;

3.4.4. (Deprecated:) K8s Master Deployment

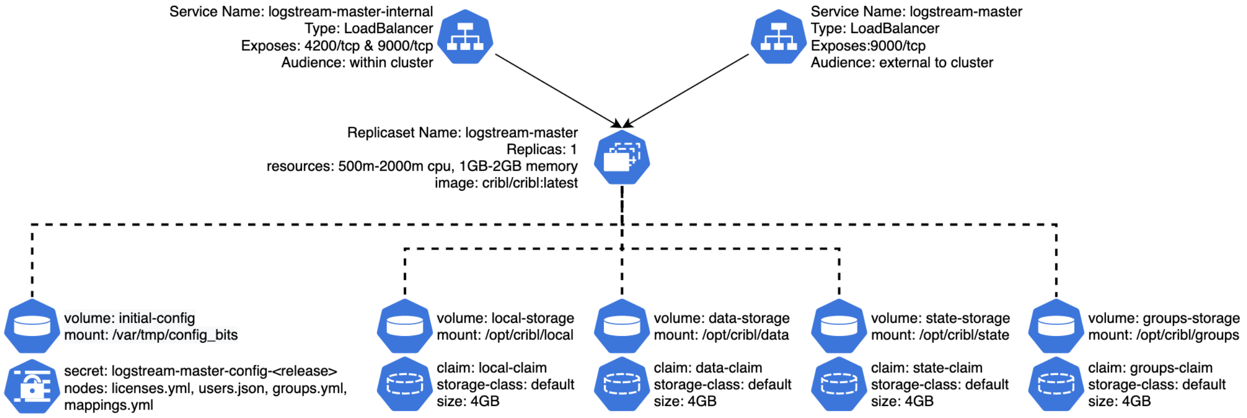
Boot a fully provisioned Leader Node via Helm

> As of Cribl Stream version 3.0.2, this chart is deprecated. Please instead see the successor K8s Leader Deployment documentation, which includes instructions for migrating an existing `logstream-master` chart to the new `logstream-leader` configuration.
>
> This document preserves legacy naming, in order to match the legacy chart's configuration.

This page outlines how to deploy a Cribl Stream Leader Node (or single instance) to AWS via Kubernetes, using a Cribl-provided Helm chart.

Deployment

As built, Cribl's chart will deploy a Master Server for Cribl Stream, consisting of a deployment, two services, and a number of persistent volumes.



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-master 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 Group/Mappings override described below.
>
> You can also use Cribl's separate logstream-workergroup chart. For details, see Kubernetes Deployment: Worker Group 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:

```
[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:

```
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-master/values.yaml
2. Save this as a local file, e.g.: `/bar/values.yaml`
3. Modify values as necessary (see Values to Override below).
4. Install your updated values to Helm, using this command: `helm install -f /bar/values.yaml`

Values to Override

This section covers the most likely values to override. To see the full scope of values available, run: `helm show values cribl/logstream-master`

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`config.adminPassword`	String	[No default]	The password you want to assign to the admin user.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
`config.token`	String	[No default]	The auth key you want to set up for Worker access. If you set this value, the Cribl Stream instance will be configured only as a Leader server for a distributed deployment. (You can also configure this later via the Cribl Stream UI, after launching the instance in single-instance mode.)
`config.license`	String	[No default]	The license for your Cribl Stream instance. If you do not set this, it will default to the Free license. You can change this in the Cribl Stream UI as well.
`config.groups`	List	[No default]	Array of Worker Group/Fleet names to configure for the Leader instance. This will create a mapping for each Group, which looks for the tag `<groupname>`, and will create the basic structure of each Group's configuration.
`config.scName`	String	<default StorageClass name>	The StorageClass name for all of the persistent volumes.
`config.rejectSelfSignedCerts`	Number	`0`	Either `0` (allow self-signed certificates) or `1` (deny self-signed certs).
`config.healthPort`	Number	`9000`	The port to use for health checks (readiness/live).
`config.healthScheme`	String	`HTTP`	The scheme to use for health checks. Supports `HTTP` or `HTTPS`.
`service.internalType`	ClusterIP	[No default]	The type to use for the `<release>-master-internal` service. In 2.4.5+, this is set to `ClusterIP` by default. If you have any Worker Groups/Fleets outside of the Kubernetes cluster where the Leader lives, you'll need to change this to `NodePort` or `LoadBalancer` to expose it outside of the cluster.
`service.externalType`	Load Balancer	[No default]	The type to use for the user-facing `<release>-master` service. If `ingress.enable` is set, this will be force-set to `NodePort`, to work with the ingress.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION

`service.ports`	Array of Maps	`- name: api`
		`  port: 9000`
		`  protocol: TCP`
		`  external: true`
		`- name: mastercomm`
		`  port: 4200`
		`  protocol: TCP`
		`  external: false`
			The ports to make available, both in the deployment and in the service. Each "map" in this list needs the following values set:
			name
			A descriptive name, identifying what the port is being used for.
			port
			The container port to be made available.
			protocol
			The protocol in use for this port (UDP or TCP).
			external
			Set to `true` to `expose` the port on the external service, or `false` to not expose it.

`service.annotations`	String	[No default]	Annotations for the service component – this is where you'll want to put load-balancer–specific configuration directives.
`criblImage.tag`	String	`latest`	The container image tag to pull from. Cribl will increment this tag per Cribl Stream version. By default, this will use a version equivalent to the chart's `appVersion` value. You can override this with `latest` to get the latest Cribl Stream version, or with a specific Cribl Stream version number (like "2.3.3").
consolidate_volumes	boolean	[No default]	If this value exists, and the `helm` command is `upgrade` , this will use the split volumes that we created in charts before 2.4 and consolidate them down to one config volume. This is a **one-time** event.

Extra Configuration Options

The links here point to configuration details on our GitHub repo.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
extraVolumeMounts	Object	[No default]	Additional volumes to mount in the container.

KEY	TYPE	DEFAULT VALUE	DESCRIPTION
extraSecretMounts	Array	[No default]	Pre-existing Secrets to mount within the container.
extraConfigmapMounts	Object	[No default]	Pre-existing ConfigMaps to mount within the container.
extraInitContainers	Object	[No default]	Additional containers to run ahead of the primary container in the pod.
securityContext.runAsUser	Number	0	User ID to run the container processes under.
securityContext.runAsGroup	Number	0	Group ID to run the container processes under.
envValueFrom	Object	[No default]	Environment variables to be exposed from the Downward API.
env	Array	[No default]	Additional static environment variables.
ingress.enable	boolean	false	Enable Ingress in front of the external service. Setting this to `true` changes the external service to type `NodePort`, and creates an ingress that connects to it.
ingress.annotations	Object	[No default]	If `ingress.enable` is set to `true`, this is where annotations to configure the specific ingress controller. (NOTE: Ingress is supported only on Kubernetes 1.19 and later clusters).

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:

```
    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: "<bucket
name>"
    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.

> More options are coming here!

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-master`:
  `helm install logstream-master cribl/logstream-master`

- To install the chart using the storage class `ebs-sc`:
  `helm install logstream-master cribl/logstream-master --set config.scName='lebs-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-master cribl/logstream-master --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 global ⚙ **Settings** (lower left) > **Licensing** UI page and accept the Free license. (The Free license allows one Worker Group.)

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 global ⚙ **Settings** (lower left) > **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-master cribl/logstream-master --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-master cribl/logstream-master --set config.groups=
{group1,group2,group3}
```

The example above will create three Worker Groups/Fleets – `group1`, `group2`, and `group3` – and a Mapping Rule for each.

Persistent Volumes

Cribl Stream 2.4.0 and above support the `$CRIBL_VOLUME_DIR` environment variable. This variable simplifies the chart's persistent-storage requirement, by specifying a path where Cribl Stream should store the persistent data.

Instead of maintaining multiple volumes (one each for `$CRIBL_HOME/{.git, data, state, local, groups, log}`), you can persist data using a single volume. `$CRIBL_VOLUME_DIR` instructs Cribl Stream where to place these persistent directories, by overriding `$CRIBL_HOME`.

Using Persistent Volumes

To use this feature, pass the `$CRIBL_VOLUME_DIR` variable to your container's environment. Make sure it points to the same value as the volume's mount point.

To start out simple, here is a minimal working example using Docker:

```
docker volume create example-volume
docker run -e CRIBL_VOLUME_DIR=/mount/point -v example-volume:/mount/point
cribl/cribl
```

For Kubernetes, as shown in the example below, you'll need to create:

- A persistent volume claim.
- An environment variable definition.
- A volume mount definition.
- A volume definition.

```
---
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: config-claim
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 20Gi

---
apiVersion: apps/v1
kind: Deployment
metadata:
  name: logstream-master
spec:
  replicas: 1
  template:
    spec:
      containers:
      - name: cribl-master
        image: cribl/cribl
        env:
        - name: CRIBL_VOLUME_DIR
          value: /mount/point
        volumeMounts:
        - name: example-volume
          mountPath: /mount/point
  volumes:
    - name: config-storage
      persistentVolumeClaim:
        claimName: config-claim
```

Notes on Persistent Volumes

- See your Kubernetes service's documentation for appropriate details on setting up persistent storage and default storage classes.
- If you use `$CRIBL_VOLUME_DIR`, you **must** set it on the Leader Node or single instance.
- Also set `$CRIBL_VOLUME_DIR` on Worker Nodes/Edge Nodes if you are using Persistent Queue on any Destination. Doing so will retain data if the container is shut down while data is still queued.
- Setting this variable for **existing** instances will leave existing data intact in the original directories. (Note that data loss is still possible if you don't persist the data by other means.)
- For more usage details, see the CLI Reference.

Upgrading Pre-2.4.0 Versions to Use Persistent Volumes

To enable a persistent volume using the `$CRIBL_VOLUME_DIR` environment variable, you'll need to upgrade any pre-2.4.0 version of Cribl Stream. In the Helm chart, we handle this via the `helm upgrade` command.

If you are upgrading from a pre-2.4 version of the chart, you'll want to set the `consolidate_volumes` value, which will create a new, larger volume, and consolidate the data from the original volumes into that volume. An `initContainer` handles the logistics. When this process completes, the `logstream-master` pod will come back up with a single consolidated volume.

> ### Back Up Your Data First
>
> While we've tested this upgrade repeatedly, differences in environments can always cause problems. Therefore, we recommend that you back up your data before running the upgrade command. This is best done with a combination of `kubectl` and `tar`:
>
> ```
> kubectl exec <pod name> -n <namespace> -- bash -c "cd /opt/cribl; tar cf -
> {state,data,local,groups}" > cribl_backup.tar
> ```
>
> This command executes the `tar`-based backup of all four volumes, and outputs it to a local `.tar` file (`cribl_backup.tar`).

Running the Upgrade

Helm makes upgrades easy. You simply need to run `helm repo update` to ensure you have the latest repo updates available, followed by `helm upgrade` to actually upgrade the containers.

For example, if you've installed the Helm charts in the `logstream` namespace, named your release `ls-master`, and set up your Helm repo according to the prerequisites section above (i.e., named it `cribl`), run the following:

```
helm repo update
helm upgrade ls-master --set consolidate_volumes=true -n logstream cribl/logstream-
master
```

Upgrade Order of Operations

While there should be no major problems running a 2.4.0 master and 2.3.4 workers, Cribl does not recommend this. Instead, upgrade the master Helm chart to 2.4.0 first, and then upgrade the workers. (For details, see Kubernetes Worker Deployment.)

Idempotency of Upgrade

The upgrade operation performs a potentially destructive action in coalescing the 4 volumes to a single volume. But that operation proceeds only if the single volume has no data on it. Once the upgrade is performed the first time, any further upgrade operations will effectively skip that coalescence operation, without causing any additional issues.

Recovering from a Failed Upgrade

If the upgrade fails, the suggested recovery path is to remov the Helm chart, reinstall it, and then run this command to restore the data from the backup:

```
cat cribl_backup.tar| kubectl -n <namespace> exec --stdin <pod name> -- bash -c "cd /opt/cribl/config-volume/; tar xf -"
```

This will restore the data into the "new" volume (which is mounted as `/opt/cribl/config-volume`). If you want to double-check that:

```
kubectl -n <namespace> exec <pod name> -- bash -c "ls -alR /opt/cribl/config-volume"
```

Preloading Configuration

The `extraConfigmapMounts` and `extraSecretMounts` options enable you to preload configuration files into the master 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 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 ../master-values.yaml -n logstream-ht cribl/logstream-master
```

You can now get the pod's name:

```
% kubectl get pods -n logstream-ht
NAME                                               READY   STATUS    RESTARTS   AGE
lsms-master-659bfccdd6-xsz67                       1/1     Running   0          52m
```

And then you can use `kubectl describe` to get the relevant environment variables:

```
% kubectl describe  pod/lsms-master-659bfccdd6-xsz67 -n logstream-ht  | egrep
"CRIBL_.*START"
CRIBL_BEFORE_START_CMD_1:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/licenses.yml
]; then mkdir -p $CRIBL_VOLUME_DIR/local/cribl ; cp
/var/tmp/config_bits/licenses.yml $CRIBL_VOLUME_DIR/local/cribl/licenses.yml; fi
CRIBL_BEFORE_START_CMD_2:      if [ ! -e $CRIBL_VOLUME_DIR/local/cribl/mappings.yml
]; then mkdir -p $CRIBL_VOLUME_DIR/local/cribl;  cp /var/tmp/config_bits/groups.yml
$CRIBL_VOLUME_DIR/local/cribl/groups.yml; cp /var/tmp/config_bits/mappings.yml
$CRIBL_VOLUME_DIR/local/cribl/mappings.yml; fi
CRIBL_AFTER_START_CMD_1:       [ ! -f $CRIBL_VOLUME_DIR/users_imported ] && sleep 20
&& cp /var/tmp/config_bits/users.json $CRIBL_VOLUME_DIR/local/cribl/auth/users.json
&& touch $CRIBL_VOLUME_DIR/users_imported
```

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/Fleet. 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}/${_time:%Y}/${_time:%m}/
      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 master 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/group1/local/cribl/jobs.yml"
```

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 directory tree exists
  cp /var/tmp/job-config/jobs.yml /opt/cribl/config-volume/groups/group1/local/cribl # copy the file
  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 [ ! -e $FLAG_FILE ]; then mkdir -p /opt/cribl/config-volume/groups/group1/local/cribl; cp /var/tmp/job-config/jobs.yml /opt/cribl/config-volume/groups/group1/local/cribl; touch $FLAG_FILE; fi"
```

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 `PersistentVolume`s and `PersistentVolumeClaim`s 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.

;

3.5. 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:

```
version: '3.5'
services:
  master:
    image: ${CRIBL_IMAGE:-cribl/cribl:latest}
    environment:
      - CRIBL_DIST_MODE=master
      - CRIBL_DIST_MASTER_URL=tcp://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_IMAGE:-cribl/cribl:latest}
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://criblmaster@master:4200
    ports:
      - 9000
```

This uses a local directory, `~/cribl-config`, as a persistent configuration store for Cribl Stream. **You must create this directory** on your host OS' filesystem before you run the `docker-compose` command.

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

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

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.

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. As of Cribl Stream v.3.5, our Docker images are built on Ubuntu 20.04.

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          STATUS
PORTS                                         NAMES
a3de9ea8f46f   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
seconds   0.0.0.0:63411->9000/tcp                       docker_workers_1
40aa687baefc   cribl/cribl:latest   "/sbin/entrypoint.sh…"   12 seconds ago   Up 10
seconds   0.0.0.0:63410->9000/tcp                       docker_workers_2
df362a65f7d1   cribl/cribl:latest   "/sbin/entrypoint.sh…"   13 seconds ago   Up 11
seconds   0.0.0.0:19000->9000/tcp, :::19000->9000/tcp   docker_master_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:

```
version: '3.5'
services:
  [...]
  workers:
    image: ${CRIBL_IMAGE:-cribl/cribl:latest}
    depends_on:
      - master
    environment:
      - CRIBL_DIST_MODE=worker
      - CRIBL_DIST_MASTER_URL=tcp://criblmaster@master:4200
    ports:
      - 9000
      - <Additional ports go here>
```

You can specify port forwarding, and TCP versus UDP protocol, in the same keys:

```
    ports:
      - 9000
      - "10060:10060/tcp"
      - "10070:10070/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_IMAGE:-cribl/cribl:latest}
```

You can explicitly pull the latest stable image with this CLI command:

```
docker pull cribl/cribl:latest
```

If you choose to keep an earlier image in production, Cribl strongly recommends that you monitor and patch vulnerabilities in the packaged OS.

;

4. Administering

4.1. Licensing

Every Cribl Stream download package ships with a Free license that allows for processing of up to 1 TB/day. This license requires sending anonymized telemetry metadata to Cribl. (For details, see Telemetry Data below).

Enterprise, Standard, and Sales Trial licenses do **not** require sending telemetry metadata, and are entitled to a defined, per-license daily ingestion volume.

This page summarizes all these license types.

Managing Licenses – Adding and Renewing

You can add and manage licenses in global ⚙ **Settings** (lower left) > **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.

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, Cribl TCP, and Cribl Stream (Deprecated) Sources does not count against your ingestion quota.
- Data sent from Cribl Edge's Cribl HTTP, Cribl TCP, and Cribl Stream (Deprecated) Destinations to Cribl Stream is counted against quota only in Cribl Edge.
- Data generated by Datagens does not count against your ingestion quota.

License Types

Cribl offers four Cribl Stream license types, summarized below.

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

Enterprise License

This is a license available for purchase.

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

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

Standard License

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

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

Standard licenses, like Enterprise licenses, support Git remote backup. Contact Cribl Sales at sales@cribl.io for more information.

Free License

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

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

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

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

Sales Trial License

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

Combining License Types

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

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

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

> **When Licenses Expire**
>
> Upon expiration of a paid license, if there is no fallback license, Cribl Stream will backpressure and block all incoming data.

Licensing in Distributed Deployments

With licenses that limit the number of Worker/Edge Node Processes, Cribl Stream will attempt to balance (or rebalance) Worker/Edge Node (threads) as evenly as possible across all licensed Worker/Edge Nodes.

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

Telemetry Data

A **Free** license or Cribl.Cloud plan 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 on `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!
```

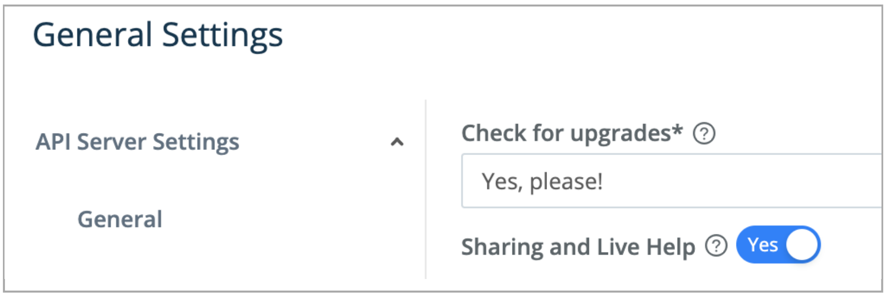
If you get a 302 response code, check whether you've omitted the URL's trailing `/`.

Disabling Telemetry and Live Help

With an Enterprise or Standard license, you have the option to disable telemetry sharing from on-prem Cribl Stream. (This option does not work on any Cribl.Cloud plan.) 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 global ⚙ **Settings** (lower left) > **System** > **General** > **Upgrade & Share Settings** > **Sharing and Live Help**. Toggle the slider to **No**.



General Settings

Check for upgrades*: Yes, please!

Sharing and Live Help: Yes

Sharing and Live Help toggle

> Disabling this setting also removes Cribl Stream's Intercom 💬 or Salesforce (live help) widget at lower right. Therefore, you will need to submit help requests, screenshots, and diag bundles through other support channels.

Metadata Shared Through Telemetry

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

- Version
- Instance's GUID
- License ID
- Earliest, Latest Time
- Number of Events In and Out, overall and by Source type and Destination type
- Number of Bytes In and Out, overall and by Source type and Destination type
- Number of Open, Closed, Active Connections
- Number of Routes
- Number of Pipelines

Licensing FAQ

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

Open Cribl Stream's global ⚙ **Settings** (lower left) > **Licensing** page to see these details.

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

Forward Cribl Internal metrics to your Metrics Destination of choice, and run a report on `cribl.total.in_bytes`.

How does Cribl enforce license limits?

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

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

- Free, and Standard licenses enforce data ingestion quotas through limits on the number of Worker Groups/Fleets and Worker/Edge Node Processes.
- Enterprise license keys turn off all enforcement.
- When an Enterprise or Standard license expires, Cribl 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 global ⚙ **Settings** (lower left) > **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.

;

4.2. 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.

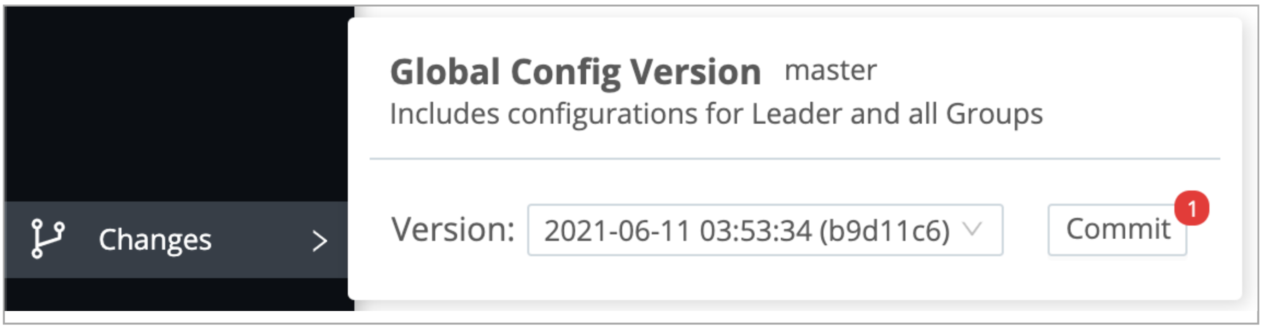
Licensing Dashboard

Even on single-instance deployments, the Monitoring > Licensing dashboard will display configuration change markers only if you have `git` installed.

Committing Changes

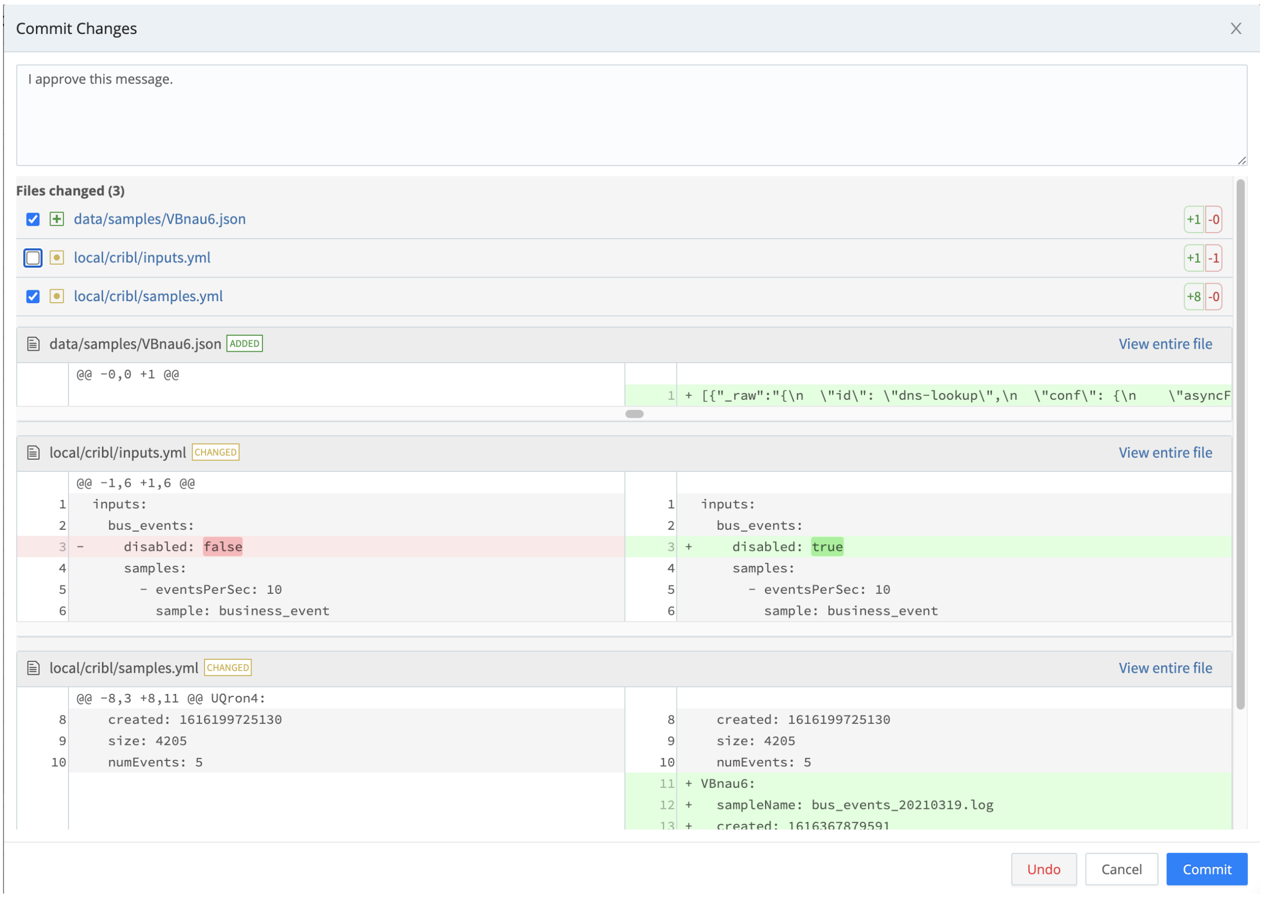
Once Git is installed, you can commit configuration changes using the `git` CLI. You can also commit changes interactively, using Cribl Stream's UI.

Pending commits have a red dot indicator, as shown below. Click **Commit** to proceed.


Changes
Global Config Version
master
Includes configurations for Leader and all Groups
Version:
2021-06-11 03:53:34 (b9d11c6)
Commit
1

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 Changes

I approve this message.

Files changed (3)

- ☑ data/samples/VBnau6.json +1 -0
- □ local/cribl/inputs.yml +1 -1
- ☑ local/cribl/samples.yml +8 -0

data/samples/VBnau6.json ADDED — View entire file

```
@@ -0,0 +1 @@
                                        1 + [{"_raw":"{\n  \"id\": \"dns-lookup\",\n  \"conf\": {\n    \"asyncF
```

local/cribl/inputs.yml CHANGED — View entire file

```
@@ -1,6 +1,6 @@
1   inputs:                              1   inputs:
2     bus_events:                        2     bus_events:
3 -     disabled: false                  3 +     disabled: true
4       samples:                         4       samples:
5         - eventsPerSec: 10             5         - eventsPerSec: 10
6           sample: business_event       6           sample: business_event
```

local/cribl/samples.yml CHANGED — View entire file

```
@@ -8,3 +8,11 @@ UQron4:
8      created: 1616199725130            8      created: 1616199725130
9      size: 4205                        9      size: 4205
10     numEvents: 5                      10     numEvents: 5
                                         11 + VBnau6:
                                         12 +   sampleName: bus_events_20210319.log
                                         13 +   created: 1616367879591
```

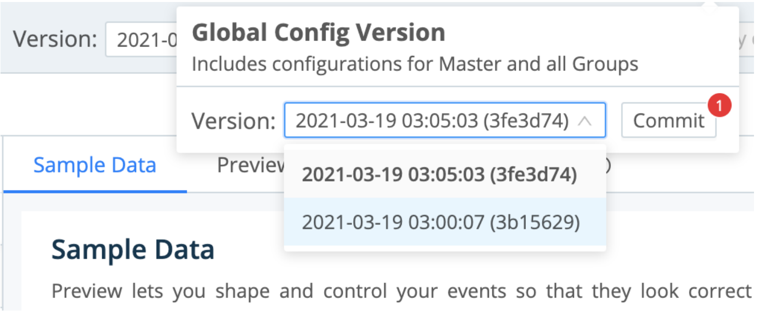
Reviewing a pending commit

When you're ready to commit to your commit, click **Commit**. Look for a **Commit successful** confirmation banner.

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.



Global Config Version

Includes configurations for Master and all Groups

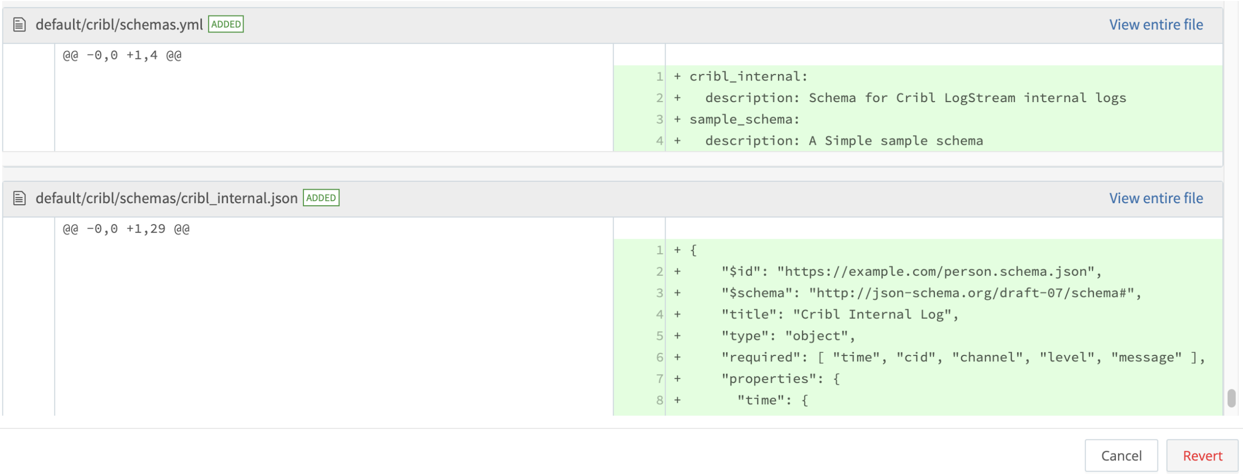
Version: 2021-03-19 03:05:03 (3fe3d74)

- 2021-03-19 03:05:03 (3fe3d74)
- 2021-03-19 03:00:07 (3b15629)

Sample Data

Preview lets you shape and control your events so that they look correct

Then, in the resulting **Commit** modal, verify the diff'ed configuration changes and click **Revert**.



default/cribl/schemas.yml ADDED — View entire file

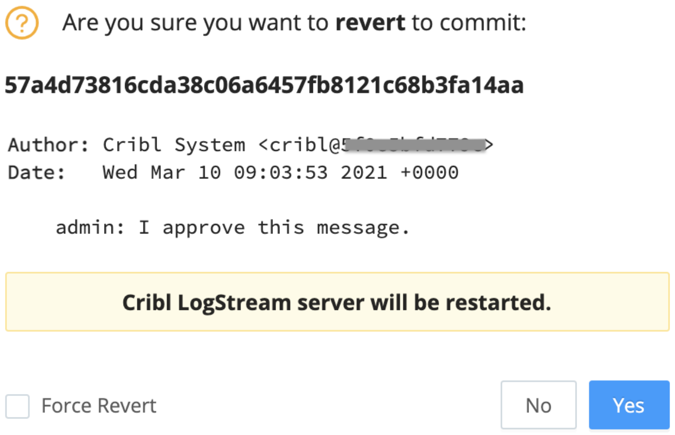
```
@@ -0,0 +1,4 @@
+ cribl_internal:
+   description: Schema for Cribl LogStream internal logs
+ sample_schema:
+   description: A Simple sample schema
```

default/cribl/schemas/cribl_internal.json ADDED — View entire file

```
@@ -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 Standalone Leader Node with Git remote push capabilities through the Cribl Stream CLI, or through the Cribl Stream UI (via global ⚙ **Settings** (lower left) > **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 Remote Repos

> 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 with a Personal Access Token over HTTPS (Recommended)

Cribl recommends connecting to a remote repo over HTTPS. The example below shows a token-based HTTPS connection to GitHub.

Example: Connecting to GitHub over HTTPS

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. Create a personal access token with **repo** scope.

3. Copy the token to your clipboard.

4. In Cribl Stream, go to global ⚙ **Settings** (lower left) > **System** > **Git Settings**.

5. Fill in the **Remote URL** field with your repo name. Use the format below:
   `https://<accesstoken>@github.com/<reponame>.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.

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 global ⚙ **Settings** (lower left) > **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`

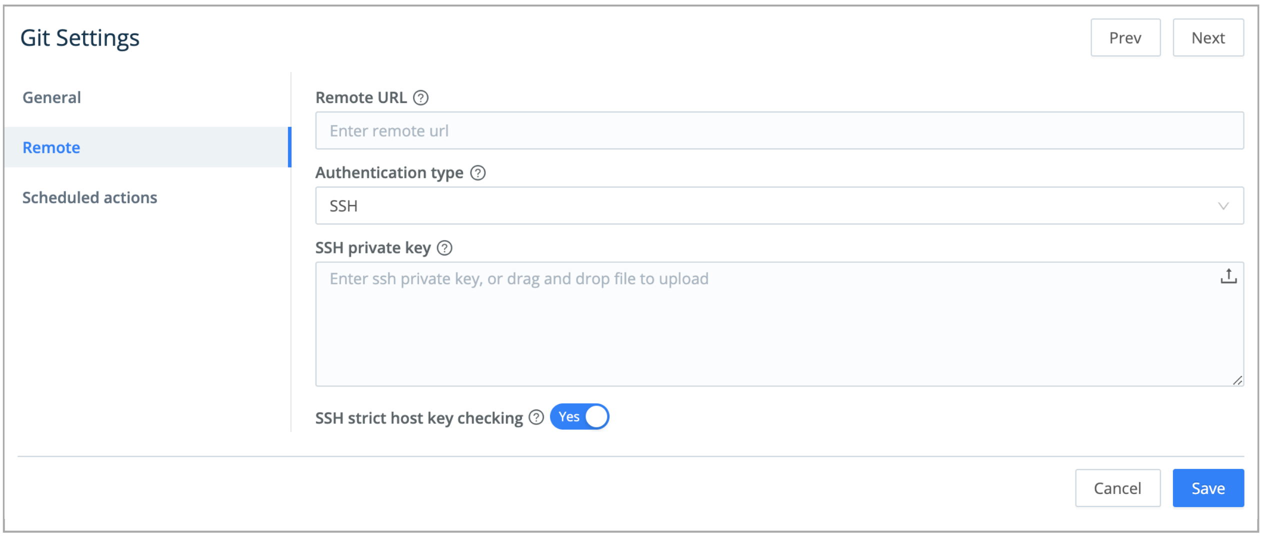
5. Paste in the **SSH private key**.

> 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 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`

GitLab's **Repository Settings > Push Rules** section includes these two settings of interest:

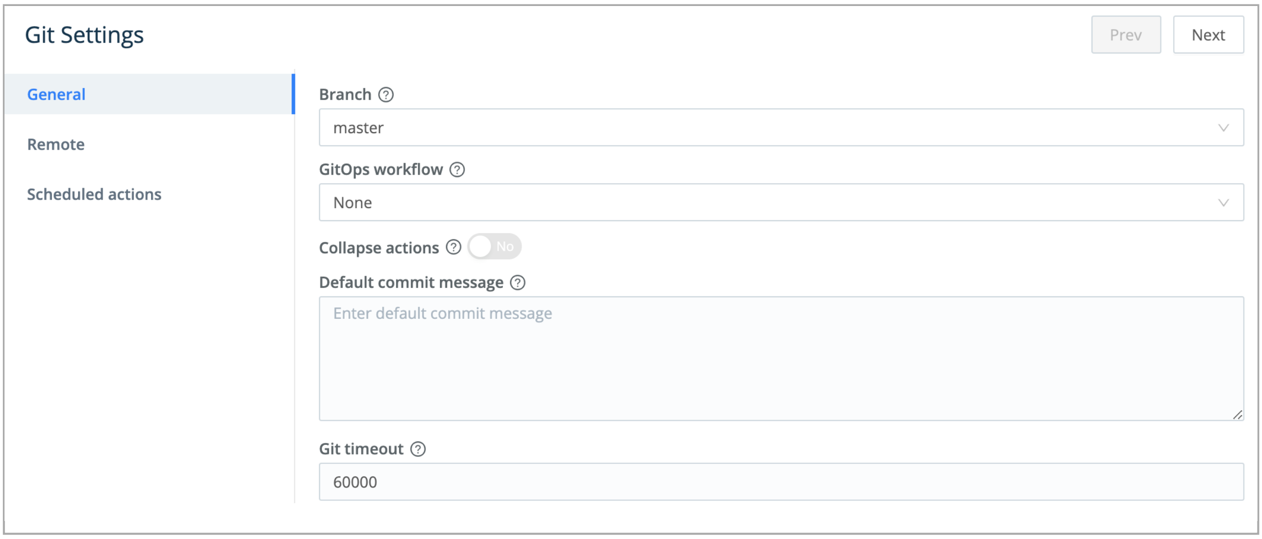
- As needed, enable **Check whether author is a GitLab user**.

- 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**: Use these drop-downs to configure GitOps.
- **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

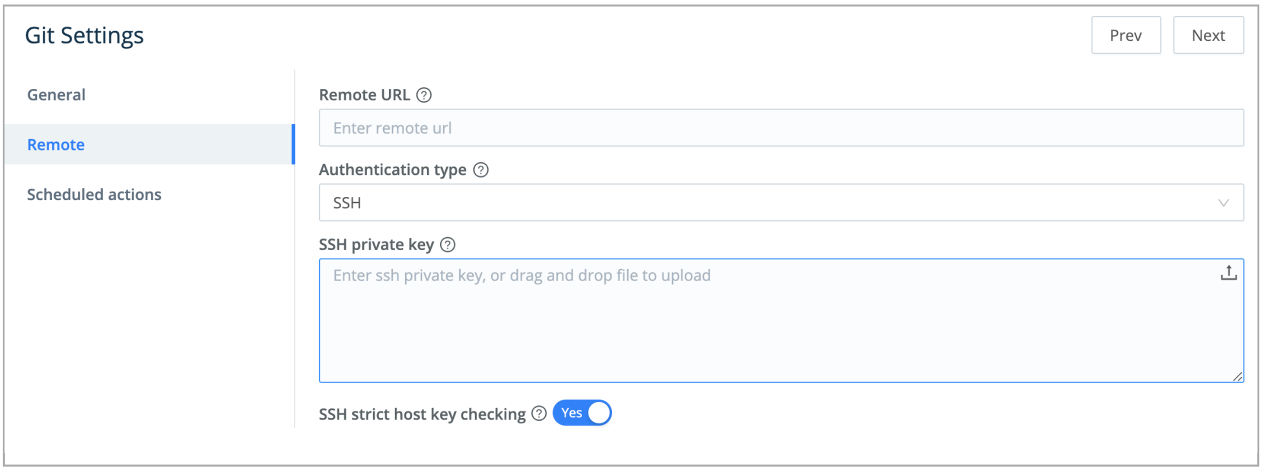
Default commit message: Enter default commit message

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).



Remote URL: Enter remote url

Authentication type: SSH

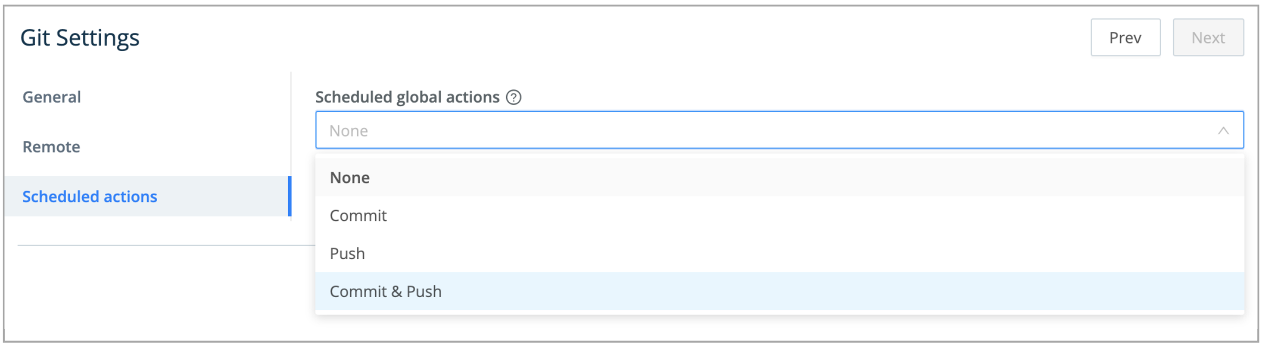
SSH private key: Enter ssh private key, or drag and drop file to upload

SSH strict host key checking: Yes

Git Authentication Type settings

> GitHub (specifically) does not support Basic authentication.

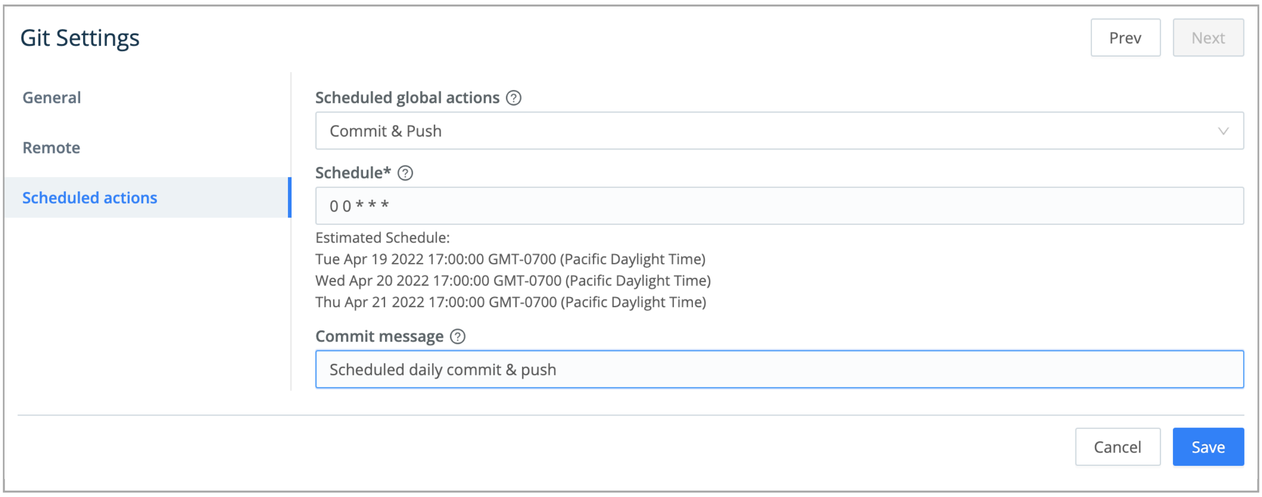
On the **Git Settings** > **Scheduled Actions** tab, you can schedule a **Commit**, **Push**, or **Commit & Push** action to occur on a predefined interval.



Scheduled global actions: None

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**.



Scheduled global actions: Commit & Push

Schedule*: 0 0 * * *

Estimated Schedule:
Tue Apr 19 2022 17:00:00 GMT-0700 (Pacific Daylight Time)
Wed Apr 20 2022 17:00:00 GMT-0700 (Pacific Daylight Time)
Thu Apr 21 2022 17:00:00 GMT-0700 (Pacific Daylight Time)

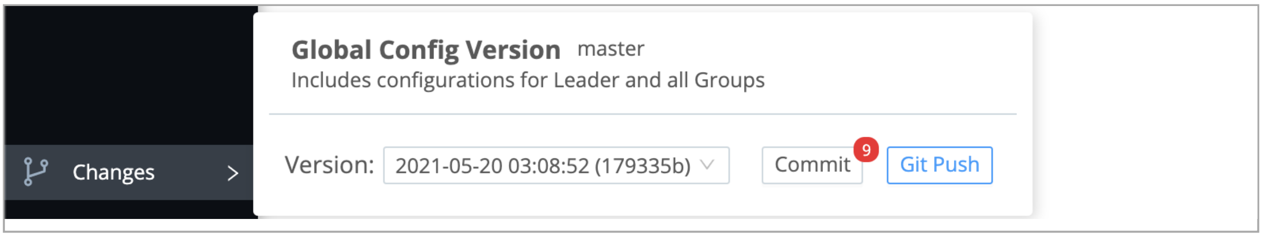
Commit message: Scheduled daily commit & push

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 **Changes** overlay. You can use this to copy committed configuration changes to your remote.



Global Config Version master
Includes configurations for Leader and all Groups

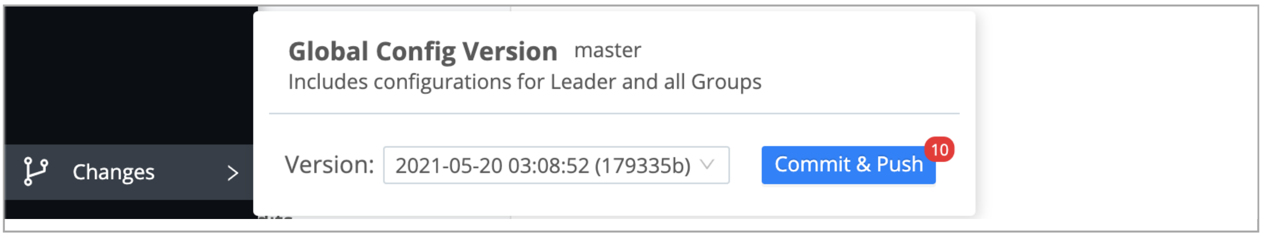
Version: 2021-05-20 03:08:52 (179335b)

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 instead see a combined **Commit & Push** button in the overlay.



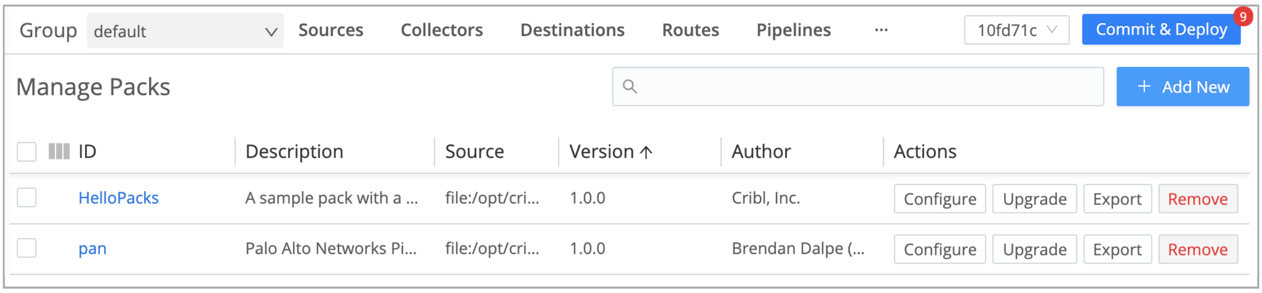
Changes

Global Config Version master
Includes configurations for Leader and all Groups

Version: 2021-05-20 03:08:52 (179335b)

Git combined actions button

On a Group's top nav, the **Collapse Actions** option will display a combined **Commit & Deploy** button at the right for the Group's config.



Manage Packs

ID	Description	Source	Version ↑	Author	Actions
HelloPacks	A sample pack with a ...	file:/opt/cri...	1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
pan	Palo Alto Networks Pi...	file:/opt/cri...	1.0.0	Brendan Dalpe (...	Configure Upgrade Export Remove

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.

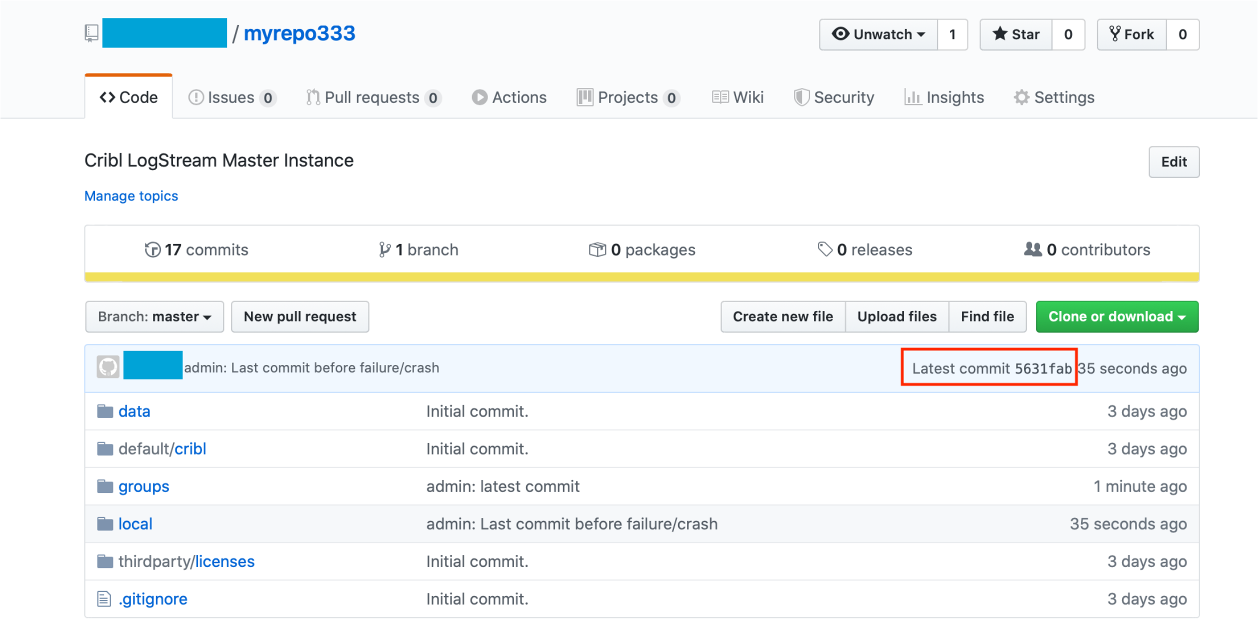
Troubleshooting Push Errors

To resolve errors commonly encountered when pushing to a remote repo, see:

- Git Push Errors
- Git Remote Repos & Trusted CAs

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 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' git
fetch origin
```

> 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`.

```
# 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

The `CRIBL SECTION` is used by Cribl Stream to define default patterns/rules that ship with every version. Do **not** add or remove any of the lines here, because Chuck Norris will easily find you!

Maslow's theory of higher needs does not apply to Chuck Norris. He has only two needs: killing people and finding people to kill. Seriously, do not remove them, as they will be overwritten on the next update. The only modifications that will survive updates are commented lines.

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/**` to `.gitignore`, then any certificate/key files stored in the default `$CRIBL_HOME/local/cribl/auth/certs/` path will be omitted from config deployments, because of a match on the `.../auth/...` subdirectory.

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","insertions":
```

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"
"https://logstream:9000/api/v1/version/commit" -d '{"message":"automation@cribl:
commit","group":"default"}'
```

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":"5b8f42a","id":"default"}],"count":1}
```

Deploy Example

Here, let's deploy the previously committed configuration to the `default` Worker Group:

```
-H "content-type: application/json" curl -X PATCH --H "Authorization: Bearer $token"
"https://logstream:9000/api/v1/master/groups/default/deploy" -d
'{"version":"abcd1234"}'
```

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/cribl/samples.yml"]
}
```

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"
"https://logstream:9000/api/v1/version/commit" -d '{"message":"automation@cribl:
commit","group":"default","files":["groups/default/data/samples/sample-
g0aT.json","groups/default/local/cribl/samples.yml"]}'
```

;

4.3. GitOps

With the GitOps features available in Cribl Stream/LogStream 3.2 and higher, you can integrate Cribl Stream configuration management with standard version-control systems and CI/CD flow: Push updates to your remote Cribl Stream via webhook, manually or via automation. This separates development configurations from production configurations, enabling you to safely build and continuously deploy your observability pipelines.

> See also our GitOps/GitHub Tutorial and Managing Stream with GitOps Sandbox.

Prerequisites

To use GitOps, you must have a customer-managed, distributed Cribl Stream deployment. (Cribl.Cloud does not currently support GitOps).

You'll need two Leader instances and a Cribl Enterprise license, which can be shared between the two instances. For questions about developer licenses in Dev environments, please contact your Cribl sales team.

You must have access to an external Git management system like GitHub, GitLab, or Bitbucket, or access to a self-hosted Git server.

Cribl recommends that you use private repositories for managing your GitOps environments, so as to prevent leaking of sensitive information.

> This page is agnostic about where you host your repos. It provides general concepts, procedures, and reference material. And it includes a setup scenario relying on Basic Auth, which might not be appropriate to your workflow.
>
> To jump straight into a tutorial written around GitHub, with authentication using SSH keys, see GitOps/GitHub Tutorial.

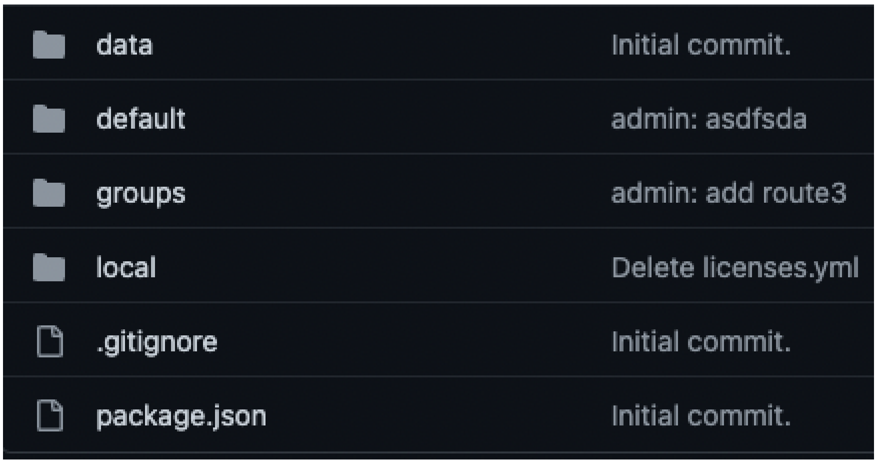
Setup: Best Practices

A Cribl Stream GitOps deployment can start with an existing repository, or you can set up an entirely new repo and start fresh from there. If you're already using an existing remote repo for your Cribl Stream deployment, Cribl recommends cloning a backup.

Create at least one branch for your production code. You will merge code into this branch and use it to update your production Cribl Stream environment.

Cribl recommends that you also create a separate dev branch for your code. This is where you will sync and work in your development environment. Once changes are validated and tested in dev, you can merge them into production.

A representative repo would have this basic structure:



data	Initial commit.
default	admin: asdfsda
groups	admin: add route3
local	Delete licenses.yml
.gitignore	Initial commit.
package.json	Initial commit.

Sample repo topography

Usage Examples

In the example below, we manually ping production Cribl Stream to update, based on a remote master branch, and to deploy the changes. You can set this up via Bearer token (used in the full example below) or via user Role.

Bearer Token

If you have your Bearer token ready, you can use it in a CLI command of this form:

```
curl -X POST "http://<leader URL or IP>:9000/api/v1/version/sync" -H "accept:
application/json" -H "Authorization: Bearer <bearer-token>" -d
"ref=master&deploy=true"`
```

Bearer tokens refresh every 60 minutes by default. To extend the lifetime of a token, go to global ⚙ **Settings** (lower left) > **System** > **General Settings** > **Advanced**, and update the **Auth-token TTL**.

Getting the Bearer Token via UI

If you need to retrieve your Bearer token before applying it, you can access it via Cribl Stream's UI, as follows:

1. From the left nav, select **API Reference**.
2. Near the top, expand the `GET /auth/groups` endpoint.
3. Click **Try it out**.
4. Click **Execute**.
5. From the displayed **Curl** field, copy the generated `curl` command.
6. Change the `GET` verb to `POST`, and execute the resulting command on your command, as shown above.

Getting the Bearer Token via CLI

If you need to retrieve your Bearer token before applying it, you can also access it in the CLI, via a series of commands of the following form. (**The final command below replaces the single `curl` command above.**)

> On your production environment's Leader, you'll need `jq`, which you can install using the command: `sudo yum install jq -y`.

Here, `<username>` and `<password>` stand for the credentials of the user getting the token:

```
mkdir -p ~/.auth

curl http://<Leader-URL-or-IP>:9000/api/v1/auth/login -H 'Content-Type:
application/json' -d "{\"username\":\"<username>\",\"password\":\"<password>\"}"
2>/dev/null | jq -r .token > ~/.auth/token

export JWT_AUTH_TOKEN=`cat ~/.auth/token`

export AUTH_HEAD="Authorization:Bearer `cat ~/.auth/token`"

curl -X POST "http://<Leader-URL-or-IP>:9000/api/v1/version/sync" -H "accept:
application/json" -H "${AUTH_HEAD}" -d "ref=master&deploy=true"
```

Here's an example with placeholders filled in. This assumes that the API is running on `localhost`, and the username and password are each the super-secret `admin`. **Here again, the final command below replaces the single `curl` command above:**

```
mkdir -p ~/.auth

curl http://localhost:9000/api/v1/auth/login -H 'Content-Type: application/json' -d
"{\"username\":\"admin\",\"password\":\"admin\"}" 2>/dev/null | jq -r .token >
~/.auth/token

export JWT_AUTH_TOKEN=`cat ~/.auth/token`

export AUTH_HEAD="Authorization:Bearer `cat ~/.auth/token`"

curl -X POST "http://localhost:9000/api/v1/version/sync" -H "accept:
application/json" -H "${AUTH_HEAD}" -d "ref=master&deploy=true"
```

User Role

Alternatively, before putting your environment into GitOps mode, create a new user and give them the `gitops` Role. This Role's only permission is to POST to the sync endpoint.

Setup Steps

This example relies on the Bearer token technique presented above.

1. Start with a production Cribl Stream, synced to an external git repository.

2. Create a new branch in the repository, called `dev`.

3. Rehydrate your dev environment from the `dev` branch.

   On the dev box's command line, run:

   ```
   CRIBL_GIT_REMOTE=https:<your_remote_repo> CRIBL_GIT_BRANCH=dev
   CRIBL_GIT_AUTH=basic CRIBL_GIT_USER=<user> CRIBL_GIT_PASSWORD=<password>
   bin/cribl start
   ```

4. Set your production environment to GitOps mode, using environment variables, as shown here:

   ```
   CRIBL_GIT_REMOTE=https:<your_remote_repo> CRIBL_GIT_BRANCH=<your prod branch,
   usually master> CRIBL_GIT_OPS=push CRIBL_GIT_AUTH=basic CRIBL_GIT_USER=<user>
   CRIBL_GIT_PASSWORD=<password> bin/cribl start
   ```

   > Your production branch will now be in read-only mode. You can push changes to production via a webhook called from your versioning system. (See the example using Git actions and our request, above.)

5. When you make changes in your dev environment, those changes will be synced with your dev repo whenever they are committed and pushed. Your production environment will be updated only when a request is sent to the production API.

6. Update the dev repo, then use PR and merge capabilities in your version-control system to merge updates into your production branch. (This will follow the workflow outlined in your version-control system of choice.)

7. Send a request to your production environment to pick up the changes on your production branch:

```
curl -X POST "http://<leader URL or IP>:9000/api/v1/version/sync" -H "accept:
application/json" -H "Authorization: Bearer <bearer token>" -d "ref=<production
branch>&deploy=true"
```

8. Alternatively, you can set up a condition in your versioning tool to automatically update production with an action. In GitHub, that action would look like this:

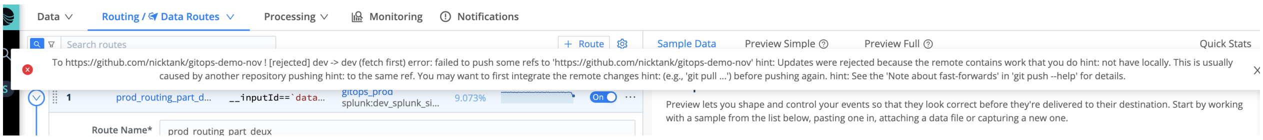
```
name: Deploy to Production
on:
  push:
    tags:
      - 'v?[0-9]+.[0-9]+.[0-9]+'
      - 'v?[0-9]+.[0-9]+.[0-9]+-[RT]C[0-9]+'
jobs:
  deployment:
    runs-on: ubuntu-latest
    steps:
      - name: deploy
        uses: satak/webrequest-action@master
        with:
          url: http://54.190.53.106:9000/api/v1/version/sync
          method: POST
          payload: '{"ref":"master","deploy":"true"}'
          headers: '{"Authorization": "Bearer ${{ secrets.BEARER_TOKEN }}"}'
```

   - This action will start whenever a new release is created in GitHub, using a tag with the naming convention `v0.0.0` or `v0.0.0-TC1` or `v0.0.0-RC1`.
   - This helps control multiple updates to production as part of a single release or deployment. This is useful when you are merging multiple changes to production, and want to push them all at once.

9. If `deploy` is set to `true` in the request to the production Cribl Stream instance, all new configs will be applied and Cribl Stream Workers will be restarted to pick up the changes.

Advanced Workflows

It's common for your repository code to get out of sync with your local version. This is especially true when first setting up a GitOps environment, where changes are being made to `.gitignore`, and to a `README` or other non-functional files in your repo. If you attempt to push from your development environment to your remote branch and you see this error:



To https://github.com/nicktank/gitops-demo-nov ! [rejected] dev -> dev (fetch first) error: failed to push some refs to 'https://github.com/nicktank/gitops-demo-nov' hint: Updates were rejected because the remote contains work that you do hint: not have locally. This is usually caused by another repository pushing hint: to the same ref. You may want to first integrate the remote changes hint: (e.g., 'git pull ...') before pushing again. hint: See the 'Note about fast-forwards' in 'git push --help' for details.

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.

Remote repo has lost sync with local config

...this means that your remote repo is ahead of your local. You can resync them via the command ,, with the following commands:

```
<cribl_home>/git fetch origin
<cribl_home>/git reset --hard origin/dev
<cribl_home>/bin/cribl restart`
```

(Replace `<cribl_home>` with your Cribl Stream directory path, and `dev` with whichever branch your environment is tracking.)

This will resync your local branch with the remote. You will lose any local changes, so before running it, stash those changes or make notes of what you will need to re-create after this reset.

Environment Variables Reference

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.

NAME	PURPOSE
CRIBL_GIT_AUTH	One of: `none`, `basic`, or `ssh`.
CRIBL_GIT_USER	Used for `basic` auth.
CRIBL_GIT_PASSWORD	Used for `basic` auth.
CRIBL_GIT_OPS	Controls which GitOps workflow to use – one of: `none`, `push`, or `pull`.

> For other environment variables available in Cribl Stream, see Distributed Deployment.
>
> To capture the name and version of the branch that GitOps is using when deploying Routes, Pipelines, or Packs, use two Eval Functions as follows:
>
> ```
> cribl_git_branch_name = C.logStreamEnv
> cribl_git_branch_version = C.confVersion
> ```

UI Options

Cribl recommends setting up GitOps using environment variables, as demonstrated above, because these are persistent. However, if you later want to undo your CLI settings, the following options are available in the UI's global ⚙ **Settings** (lower left) > **System** > **Git Settings**:

Branch drop-down: Select a single branch to track in your remote repo.

GitOps workflow drop-down provides these options for the branch selected above:

- **None**: Do not automate Cribl Stream's configuration management. (This switches off GitOps.)
- **Push**: Sync with remote repo via POST requests to `/api/v1/version/sync endpoint`.

Security

You can maintain unique encryption keys, passwords, and tokens per environment, without sharing them across (e.g.) dev vs. prod environments. Do so by setting unique environment variables per environment.

Going Further

Because you have configured separate environments for development versus production, you can create Routes, Functions, or Pipelines that behave differently in these different environments.

For this purpose, you can use the `C.env.CRIBL_GIT_BRANCH` environment variable. Its value is the name of the Git branch you're on. For any Destination, `C.env.CRIBL_GIT_BRANCH` corresponds to the **Advanced Settings** > **Environment** setting.

You can also automate GitOps in various ways. E.g., you could automate your production environment's syncing with a GitHub repo's `prod` branch, using GitHub Actions.

Known Issues

- CRIBL-6961 When the GitOps `Push` workflow has placed the UI in read-only mode, the commit UI displays a **Revert** button, even though reverting changes is not currently supported. Pressing the button will simply trigger an error message.
- CRIBL-6736 Bootstrapping a deployment from an existing repo using an SSH key works only if no passphrase is required.
- For other current limitations, please search our Known Issues topic on "GitOps".

;

4.3.1. GitOps/GitHub Tutorial

This tutorial shows one of many possible ways to set up Cribl Stream GitOps, relying on a GitHub repo with SSH authentication.

If you are using GitLab, Bitbucket, or a self-hosted Git server, you can adapt the GitHub examples here, or you can rely on our generic GitOps page.

> See also our Managing Stream with GitOps Sandbox.

Prerequisites

To use GitOps, you must have:

- A customer-managed, distributed Cribl Stream deployment. (Cribl.Cloud does not currently support GitOps).
- Two Leader instances.
- A Cribl Enterprise license, which can be shared between the two Leaders.
- A GitHub account. (You can create one here.)

Setting Up Your Deployment for GitOps

In your distributed Cribl Stream deployment, set up two separate environments: one for Development and one for Production. This way, you can do your work in the Development environment, validate and test your changes there, and then merge the changes into Production. For safety, the Production environment will be read-only.

In **both** environments:

- On the Leader, install `git` using the command:
  ```
  sudo yum install git -y
  ```
- Cribl strongly recommends that you use systemd, as we do throughout this tutorial. (If not using systemd, use environment variables instead of `systemctl` commands.)
- Cribl strongly recommends that you use SSH authentication, as we do throughout this tutorial.

In the Production environment:

- On the Leader, install `jq` using the command:
  `sudo yum install jq -y`

Creating SSH Public and Private Keys

Starting on the command line:

1. Generate public and private SSH keys, specifying Ed25519 as the key type:

   ```
   ssh-keygen -t ed25519 -C "your_email@example.com"
   ```

2. Validate that the `/root/.ssh/` directory contains both of the newly generated keys:
   - The private key that Cribl Stream will use: `id_ed25519`.
   - The public key that GitHub will use: `id_ed25519.pub`.

3. Copy the public SSH key to your clipboard.

Next, sign in to your GitHub account, and on the SSH and GPG keys page, enter the public SSH key:

1. Click **New SSH key**.
2. Paste the key in the **Key** field.
3. Click **Add SSH key**.

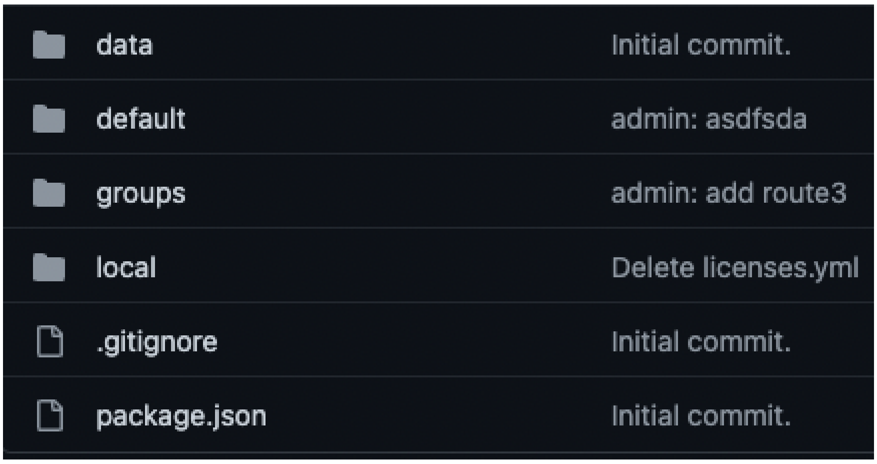
Creating a Private GitHub Repo

On your GitHub profile page:

1. Click **Go to your personal profile**.
2. On the resulting page,click the **Repositories** tab.
3. Click **New** to create a new repository.

Name the repo `cribl`, and leave it private (the default). The repo's default branch will be named `main`, and we'll assume that name in this tutorial. (Older repos' default branch might instead be named `master`.)

A representative repo would have this basic structure:



data	Initial commit.
default	admin: asdfsda
groups	admin: add route3
local	Delete licenses.yml
.gitignore	Initial commit.
package.json	Initial commit.

Sample repo topography

Configuring Your Production Leader

> You can optionally create a new user with the `gitops` Role. This Role's only permission is to `POST` to the `sync` endpoint. If you choose to do this, the **Remote URL** that you enter below should include this username.

In Cribl Stream, in the Production Leader's, select global ⚙ **Settings** (lower left) > **Git Settings**. Then apply the following settings.

General Tab

Leave **Branch** set to `main` (the only branch created thus far), and **GitOps Workflow** set to `None`.

Remote Tab

- In **Remote URL**, enter the URL for your private Git repo, e.g.: `git@github.com:<your_Git_username>/cribl.git`.
- From the **Authentication type** drop-down, select `SSH`.
- In **SSH private key**, copy and paste the private key (`/root/.ssh/id_ed25519`).
- Toggle **SSH strict host key checking** to `No`.

Click **Save**, then **Commit** and **Git Push**. You should now see your commit in the `main` branch of your private GitHub repo.

In a terminal on the Production Leader, run as root:

```
sudo su -
```

Add `systemctl` overrides to set environment variables that will make the Production Leader read-only on startup, and set the branch to `prod`:

1. Use `systemctl` to open the `cribl.service` file in an editor:

   ```
   systemctl edit cribl.service
   ```

2. When prompted, add the following overrides:

   ```
   [Service]
   Environment=CRIBL_GIT_OPS=push
   Environment=CRIBL_GIT_BRANCH=prod
   ```

3. Restart:

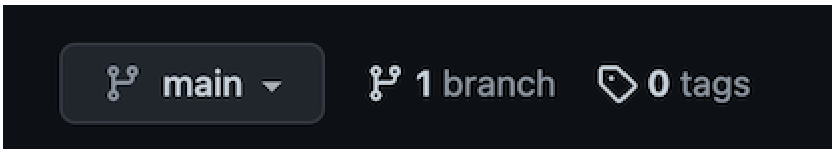
   ```
   systemctl daemon-reload && systemctl restart cribl
   ```

> Once the system has restarted, the `CRIBL_GIT_OPS=push` override will be in effect, meaning that your Production environment will be in read-only mode.

Setting Up the GitHub Branches

In your GitHub repo:

1. Click the branches tab (if there's only a `main` branch in your repo, the tab will be labeled **1 branch**).



The branches tab

2. On the `main` branch's row, click the pencil button at far right.
3. Rename `main` to `prod`.
4. From the branches drop-down (which should now say `prod`), create a new branch named `dev`.

Pointing Your Development Leader to Its Branch

On Cribl Stream's Development Leader, select global ⚙ **Settings** (lower left) > **Git Settings**. Then apply the following settings.

Remote Tab

- In **Remote URL**, enter the URL for your private Git repo, e.g.: `git@github.com:<your_Git_username>/cribl.git`.
- From the **Authentication type** drop-down, select `SSH`.
- In **SSH private key**, copy and paste the private key (`/root/.ssh/id_ed25519`).
- Toggle **SSH strict host key checking** to `No`.
- Click **Save**.

General Tab

- Set **Branch** to `dev`.
- Set **GitOps Workflow** to `None`.

Click **Save**, then restart the Development Leader, which will now be able to write to GitHub.

Working in the Development Environment

Your goal is to make changes in the Development environment, and later propagate them to Production. To demonstrate this, try adding a Route, as a minimal unit of work.

1. On Cribl Stream's Development Leader, navigate to **Routing** > **Data Routes**.

2. Click **+ Route**.

3. Configure the new Route as follows:

NAME	FILTER	PIPELINE/OUTPUT
`test-gitops`	`true`	`devnull:devnull`

4. Click **Commit/Deploy**.

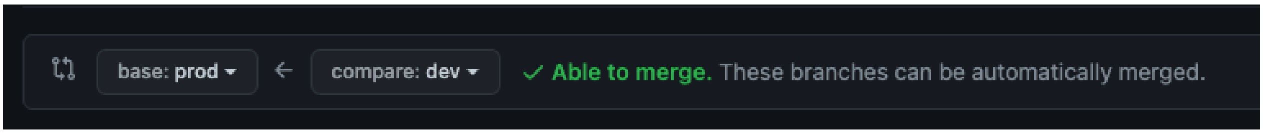
5. Click **Commit** and **Git Push**.

This will send the configurations to your `dev` branch on GitHub.

Merging Changes from `dev` to `prod`

On your GitHub `dev` branch, you should see the new `route.yml` appearing within a subdirectory of the `local` directory.

1. Click **Compare & pull request**. Be sure that the `compare` drop-down says `dev`, and the `base` drop-down says `prod`:



✓ 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 propose to `merge 1 commit into prod from dev`.
4. Assuming that GitHub does not detect any conflicts, click **Merge pull request**, then **Confirm merge**.

If the merge is successful, GitHub will display a confirmation message. You should now see the new `route.yml` in both the `dev` and `prod` branches.

Syncing Your Production Leader to GitHub

Now, to make your Production environment pick up the changes from the `prod` branch, you'll send an HTTP request from the Production Leader to the Cribl API.

Adapt the following request, substituting your Production Leader's IP address and admin password for the placeholders.

```
    curl -X POST "http://<Production_Leader_IP>:9000/api/v1/version/sync" \
    -H "accept: application/json" \
    -H "Authorization: Bearer $(curl
http://<Production_Leader_IP>:9000/api/v1/auth/login \
    -H 'Content-Type: application/json' \
    -d "{\"username\":\"admin\",\"password\":\"<Production_Leader_password>\"}"
2>/dev/null | jq -r .token)" \
    -d "ref=prod&deploy=true"
```

> Nested within the request to `/version/sync` is another request to `/auth/login`. The nested request obtains the Bearer token required by the parent request. For more about Bearer tokens, see this section.

You should now see the changes you made in your Development environment (in this example, the new Route) in the Production environment, too.

There's much more you can do with GitOps. To learn about `C.env.CRIBL_GIT_BRANCH` and GitHub Actions, see Going Further.

;

4.4. Persistent Queues

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

Persistent queues are implemented:

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

Persistent Queues Supplement In-Memory Queues

Persistent queues trigger differently on the Destination versus Source side.

Destination Side

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

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

Source Side

In Cribl Stream 3.4 and later, Push Sources' config modals also provide a PQ option. When enabled, 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.

Life Without PQ

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

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

PQ + FIFO = Durability

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

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

Source and/or Destination PQ?

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

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

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

Persistent Queue Details and Constraints

Persistent queues are:

- Available on Push Sources.
- Available on the output side (i.e., after processing) of all streaming Destinations, with these exceptions: Syslog and Graphite (when you select UDP as the outbound protocol) and SNMP Trap.

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

Configuring Persistent Queueing

Persistent Queueing is configured individually for each Source and Destination that supports it. To enable persistent queueing:

- Go to a Source's configuration modal, select the **Persistent Queue Settings** left tab, and toggle the **Enable Persistent Queue** slider to `Yes`.
- Go to a Destination's configuration modal, and set the **Backpressure Behavior** control to `Persistent Queueing`.

These selections expose the following additional controls.

Source-Side PQ Only

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `42`.

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

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

> If you enable **Compression** and also enter a **Max queue size** value, set a value higher than the volume's total available disk space (disregarding compression). This will maximize queue saturation and minimize data loss. For details, see Known Issues.

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.

Destination-Side PQ Only

Queue-full behavior: Determines whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to delete the files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

> #### Minimum Free Disk Space
>
> For queuing to operate properly, you must provide sufficient disk space. You configure the minimum disk space in global ⚙ **Settings** (lower left) > **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 5GB. Be sure to set this on your Worker Nodes (rather than on the Leader Node) when in distributed mode.

Persistent Queue Support by Destination Type

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

HTTP-Based Destinations

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

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

HTTP `400` response errors will not engage PQ, and Cribl 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:

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

TCP Load-Balanced Destinations

TCP load-balanced Destinations can be configured with one or multiple receivers. If one or more receivers go down, Cribl 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-and UDP Destinations do not support PQ. These include:

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

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

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

Other Destinations

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

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

Getting Notified about PQ Status

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

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

;

4.5. Access Management

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

- Authentication: Authenticating users in Cribl Stream.
- Local Users: Creating and managing users and their permissions.
- Roles: Managing roles and policies to assign to users.

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

;

4.5.1. Authentication

User authentication in Cribl Stream

Cribl Stream supports **local**, **Splunk**, **LDAP**, and **SSO/OpenID Connect** authentication methods, depending on license type.

Local Authentication

To set up local authentication, navigate to global ⚙ **Settings** (lower left) > **Access Management > Authentication** and select **Local**.

You can then manage users through the global ⚙ **Settings** (lower left) > **Access Management > Local Users** UI. All changes made to users are persisted in a file located at `$CRIBL_HOME/local/cribl/auth/users.json`.

This is the line format, and note that both usernames and passwords are case-sensitive:

```
{"username":"user","first":"Elvis","last":"Bath","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

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":"Elvis","last":"Bath","disabled":"false",
"passwd":"Yrt0MOD1w8OzyMYB8WMcEleOtYESMwZw2qIZyTvueOE"}
```

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

```
{"username":"user","first":"Elvis","last":"Bath","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/Edge Node, you set and push a new password to the Worker Group/Edge Fleet. Here's how, in the Leader Node's UI:

1. From the left nav, select **Groups**.
2. Select the desired Worker Group/Fleet.
3. From the Group's top nav, select **Settings** (upper right).
4. Select **Local Users**, then expand the desired user.
5. Update the **Password** field 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.

Authentication Controls

You can customize authentication behavior at global ⚙ **Settings** (lower left) > **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).

- **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

Below are configuration details for the following external authentication providers:

- Splunk Authentication
- LDAP Authentication
- SSO/OpenID Connect Authentication

> All of these 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.

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.

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 global ⚙ **Settings** (lower left) > **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`.

To prevent lockout, Cribl strongly recommends enabling **Fallback on fatal error** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

The Splunk search head does not need to be locally installed on the Cribl Stream instance. See also Role Mapping below.

LDAP Authentication

You can set up LDAP authentication as follows:

Navigate to global ⚙ **Settings** (lower left) > **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 finding user. Optional. Example: `(&(group=admin)(!(department=123*)))`
- **Group search base**,
  Group search filter,
  Group member field,
  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`.

> To prevent lockout, Cribl strongly recommends enabling **Fallback on fatal error** until you're certain that external auth is working as intended. If you do get locked out, see Manual Password Replacement.

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 search filter**: LDAP search filter to apply when finding group, 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`).

> See also Role Mapping below.

SSO/OpenID Connect Authentication

Cribl Stream supports SSO/OpenID user authentication (login/password) and authorization (user's group membership, which you can map to Cribl Roles). Using OpenID will change the default `Log in` button on the login page to a button labeled `Log in with <provider>` which redirects to the specified provider. Set this up as follows:

Navigate to global ⚙ **Settings** (lower left) > **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**: The Audience from provider configuration. This will be the base URL, e.g.: `https://master.yourDomain.com:9000` for a distributed environment.

  > For distributed environments with a second Leader configured, modify the **Audience** field to point to the load balancer instead of the Leader Node.

- **Client ID**: The `client_id` from provider configuration.

- **Client secret**: The `client_secret` from provider configuration.

- **Scope**: Space-separated list of authentication scopes. The default list is: `openid profile email`. If you populate the **User info URL** field, you must add `groups` to this list.

- **Authentication URL**: The full path to the provider's authentication endpoint. Be sure to configure the callback URL at the provider as `<masterServerFQDN>:9000/api/v1/auth/authorization-code/callback`, e.g.: `https://master.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. E.g.: `name.startsWith("someUser") && email.endsWith("domain.com")`

- **Group name field**: Field in the **User info URL** response (if configured); otherwise, `id_token` that contains the user groups. Defaults to `groups`.

- **Allow local auth**: Toggle to `Yes` to also users to log in using Cribl 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 (e.g., `https://master.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.

> See also Role Mapping below.
>
> 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.

Role Mapping

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 on the left, and select the corresponding Cribl Stream Role on the right drop-down list. Click **+ Add Mapping** to add more rows.

;

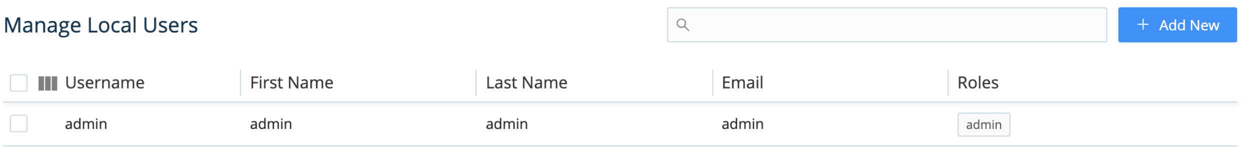
4.5.2. Local Users

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 – or in single-instance deployments (Edge, Stream). – you manage users by selecting global ⚙ **Settings** (lower left) > **Access Management** > **Local Users**.

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



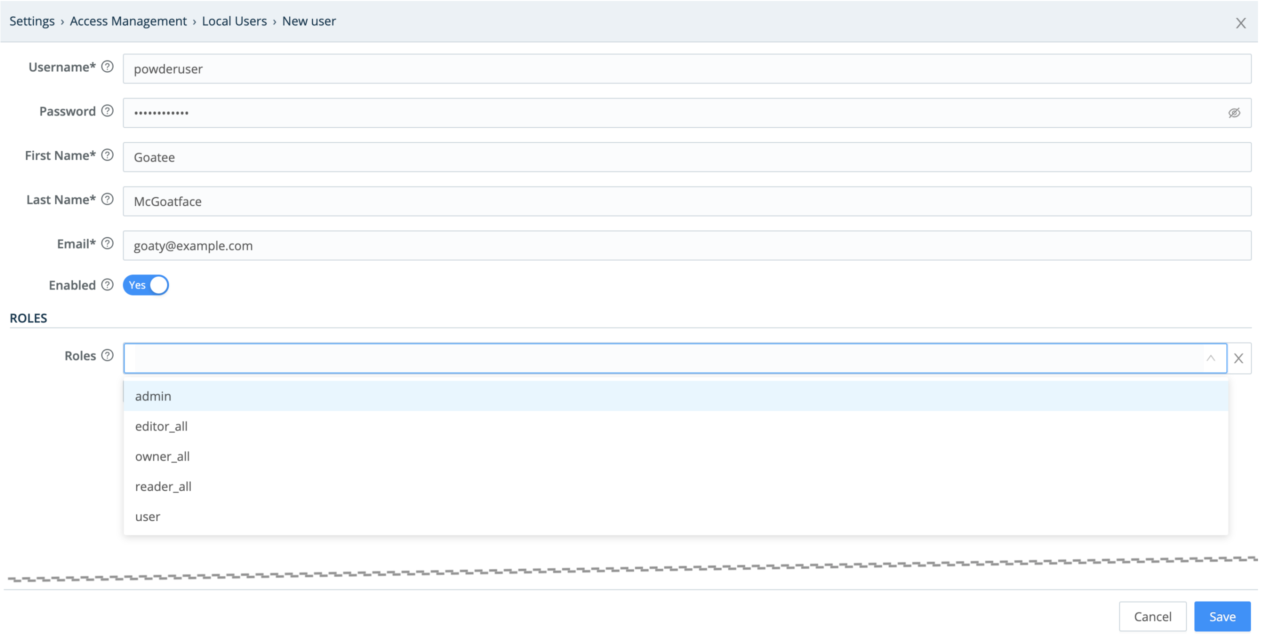
Username	First Name	Last Name	Email	Roles
admin	admin	admin	admin	admin

Managing users

To create a new Cribl Stream user, click **+ Add New**. 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 the **Enabled** slider to `No`.



Settings › Access Management › Local Users › New user

Username*	powderuser
Password	••••••••••••
First Name*	Goatee
Last Name*	McGoatface
Email*	goaty@example.com
Enabled	Yes

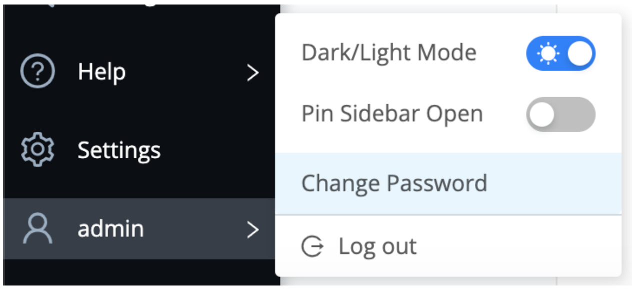
ROLES

Roles

- admin
- editor_all
- owner_all
- reader_all
- user

Entering and saving a user's credentials

In Cribl Stream 3.1 and above, logged-in users can change their own Cribl Stream passwords via the User Settings fly-out at the UI's lower left. This fly-out also provides user-specific options to customize the UI.



Self-serve password changes

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 in global ⚙ **Settings** (lower left) > **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 global ⚙ **Settings** (lower left) > **General Settings > API Server Settings > Advanced > Logout on roles change.**

;

4.5.3. 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 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.

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.

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

Cribl 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:

NAME	DESCRIPTION
admin	Superusers – authorized to do anything and everything in the system.
owner_all	Read/write access to (and Deploy permission on) all Worker Groups/Fleets.
editor_all	Read/write access to all Worker Groups/Fleets.
reader_all	Read-only access to all Worker Groups/Fleets.
collect_all	Ability to run existing collection jobs on all Worker Groups/Fleets.

NAME	DESCRIPTION
gitops	Ability to sync the Cribl Stream deployment to a remote Git repository.
notification_admin	Read/write access to all Notifications.
user	Default role that gets only a home/landing page to authenticate. This is a fallback for users who have not yet been assigned a higher role by an admin.

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

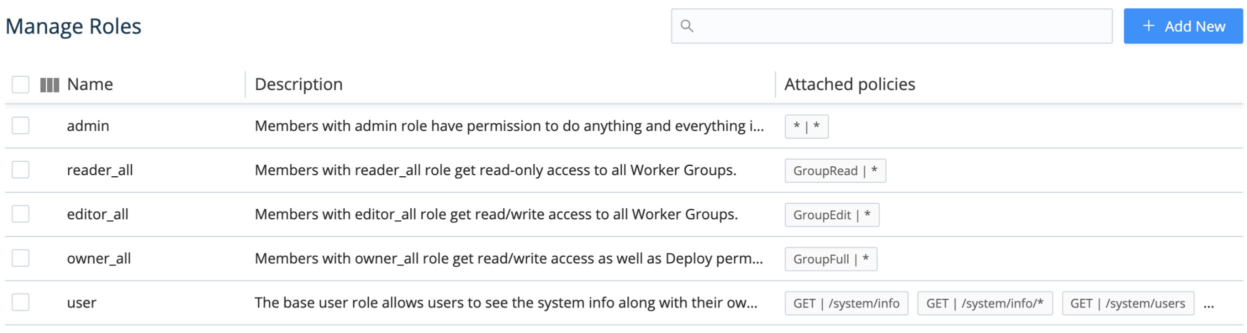
Initial Installation or Upgrade

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 global ⚙ **Settings** (lower left) > **Access Management** > **Roles**.

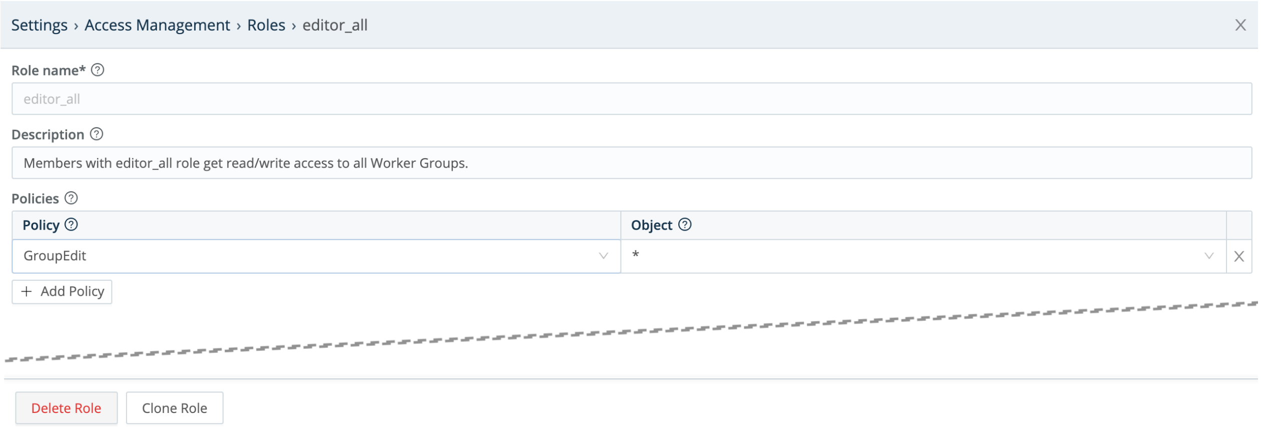


Manage Roles

□ 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 **+ Add New** 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.

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.)

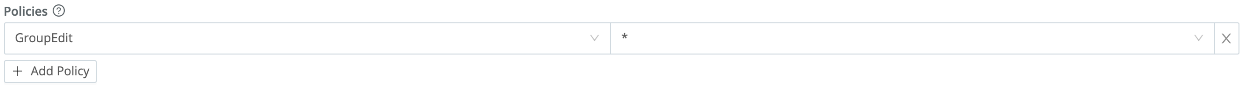
Clone Role: Opens a **New role** version of the modal, duplicating the **Description** and **Policies** of the Role you started with.

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

Adding and Modifying 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:

NAME	DESCRIPTION
`GroupRead`	The most basic Worker Group/Fleet-level permission. Enables users to view a Worker Group/Fleet and/or its configuration.
`GroupEdit`	Building on `GroupRead` , grants the ability to also change and commit a Worker Group/Fleet's configuration.
`GroupFull`	Building on `GroupEdit` , grants the ability to also deploy a Worker Group/Fleet.
`GroupCollect`	Grants the ability to run Collectors on a Worker Group/Fleet.
`*` (wildcard)	Grants **all** permissions on associated objects.

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:

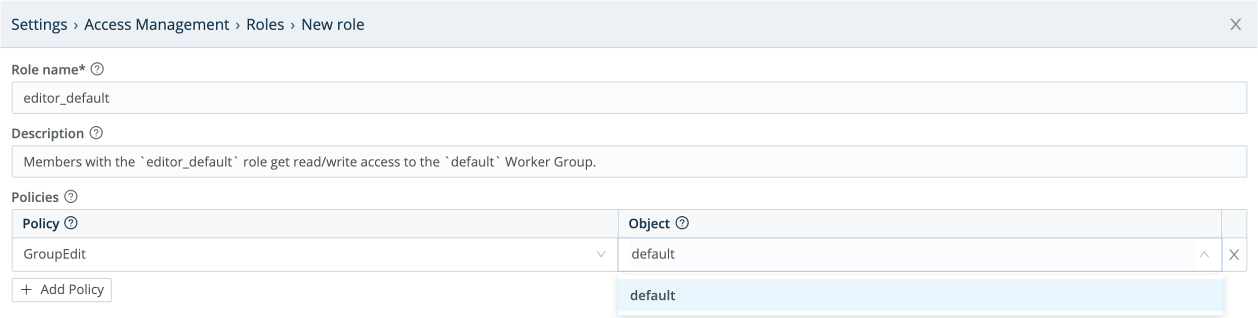
- `Worker Group <id>`
- `NewGroup2`
- `default` (Worker Group)
- `*` (all Worker Groups)

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.



Settings › Access Management › Roles › New role

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 global **Settings** (lower left) > **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.

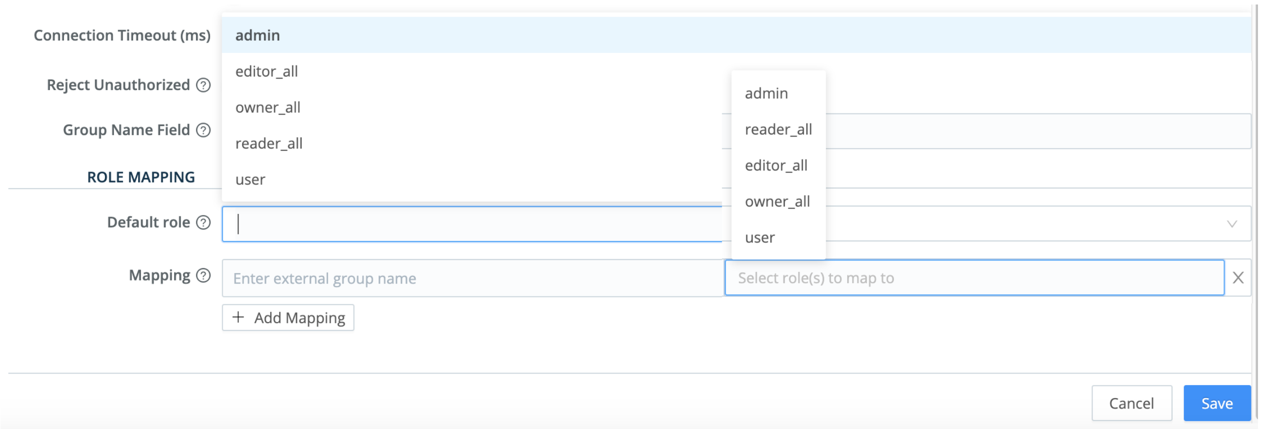
By default, Cribl Stream will log out a user upon a change in their assigned Roles. You can defeat this behavior at global ⚙ **Settings** (lower left) > **General Settings > API Server Settings > Advanced > Logout on roles change.**

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:

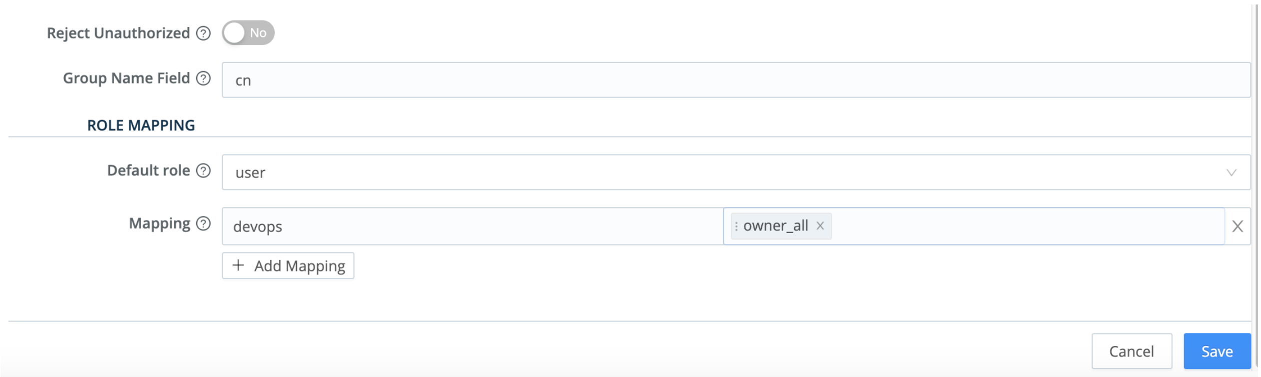
1. Select global ⚙ **Settings** (lower left) > **Access Management** > **Authentication**.
2. From the **Type** drop-down, select **LDAP**, **Splunk**, or **OpenID Connect**, according to your needs.
3. On the resulting **Authentication Settings** page, configure your identity provider's connection and 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

;

4.6. Securing

4.6.1. Securing Cribl Stream

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.

> In a single-instance deployment (Edge, Stream), wherever this page refers to a Worker Group, the equivalent left-nav link is labeled `Configure`.

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 Worker Node UI access distributed deployment (Edge, Stream) option, so that administrators will still be able to tunnel through to any Worker Nodes' UI from the Leader. This is also a prerequisite for Connecting Workers to Leader Securely.

1. Select a Group.
2. Open Group ⚙ **Settings** (top right).
3. Navigate to **Settings** > **General Settings** > **API Server Settings** > **Advanced**.
4. Toggle the **Disable UI Access** slider to `Yes`.
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 Streamexpects certificates and keys to be formatted in privacy-enhanced mail (`.pem`) format.

In the Cribl Stream UI, you can configure the cert via global ⚙ **Settings** (lower left) > **Security > Certificates**. You can configure the **key** via:

- Global ⚙ **Settings** (lower left) > **Security > Encryption Keys** single-instance deployments (Edge, Stream), or
- **Groups** > `<group-name>` > **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

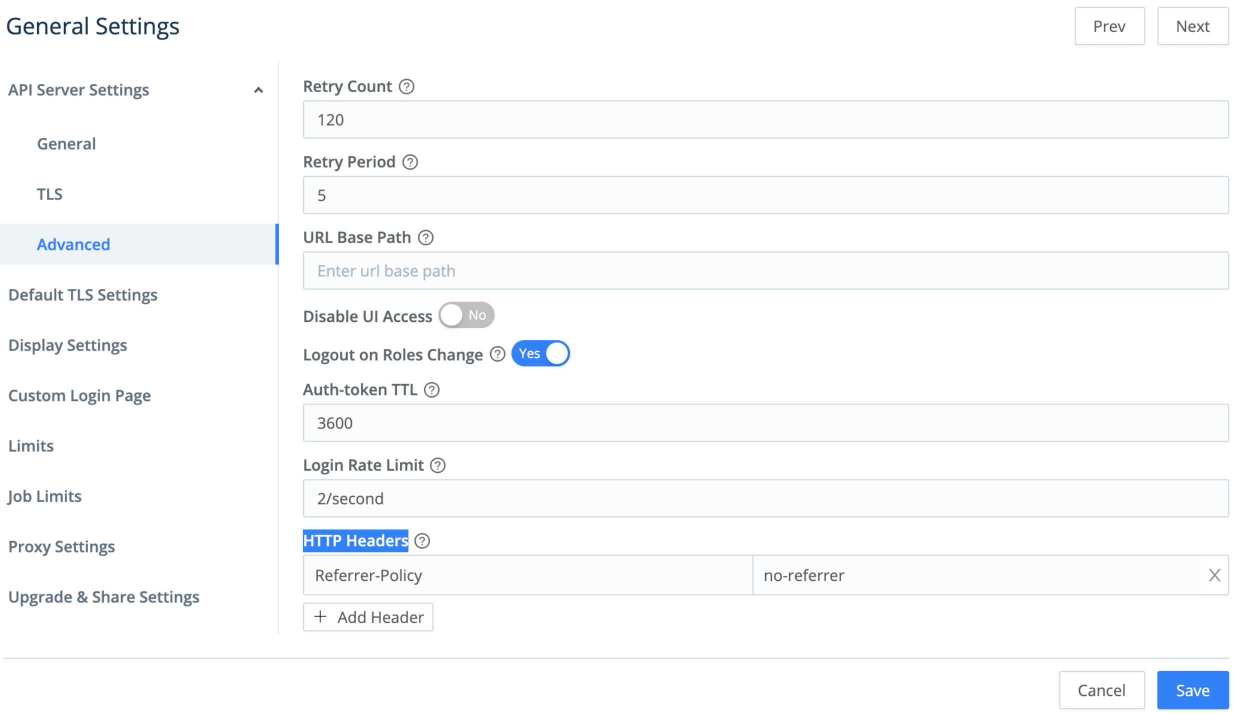
```yaml
api:
  host: 0.0.0.0
  port: 9000
  disabled : false
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
...
```

> 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 global ⚙ **Settings** (lower left) > **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

Disable UI Access: No

Logout on Roles Change: Yes

Auth-token TTL: 3600

Login Rate Limit: 2/second

HTTP Headers: Referrer-Policy | no-referrer

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	————	————	————	————	————

TRAFFIC TYPE	TLS CLIENT	TLS SERVER	ENCRYPTION	CERT AUTH	CN* CHECK
Local*	Browser	Cribl Stream	Default Disabled	N/A	N/A
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 global ⚙ **Settings** (lower left) > **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_DHE_RSA_WITH_AES_256_GCM_SHA384`
- `TLS_DHE_RSA_WITH_AES_128_GCM_SHA256`

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.

Accessing Keys

- In a single-instance deployment, select global ⚙ **Settings** (lower left) > **Security > Encryption Keys**.
- In a distributed deployment with one Worker Group, select **Configure > Settings > Security > Encryption Keys**.
- In a distributed deployment with multiple Worker Groups, keys are managed per Worker Group. Select **Groups >** `<group-name>` **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 **+ Add New** 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, the only option supported here is `aes-256-cbc`.

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.

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 global ⚙ **Settings** (lower left) > **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 **+ Add New** 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.

KMS Configuration

Cribl Stream's Key Management Service maintains the keys that Cribl Stream uses to encrypt secrets on Worker Groups and Worker Nodes.

In a single-instance deployment, the KMS is configured at global ⚙ **Settings** (lower left) > **Security > KMS**. In a distributed deployment, the Leader's KMS is configured at the same global location, while additional KMS configs for each Worker Group are available at the Group's ⚙ **Settings** (upper right) > **Security > KMS** page.

In versions 3.0 and above, administrators with a Cribl Stream Enterprise or Standard license can integrate an external KMS provider. In all, the **KMS Provider** drop-down currently provides these options:

- Cribl Stream Internal KMS: The **only** option with a Free license. 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 a 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 global ⚙ **Settings** (lower left) > **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).

Authentication

Auth provider: The method for authenticating requests to 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

Use the **Authentication method** buttons to select one of the following AWS methods:

- **Auto**: 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.

- **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 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.

Assume Role

This section is displayed for all AWS IAM authentication methods.

Enable for Vault Auth: Toggle to `Yes` if you want to use your Assume Role credentials to access Vault authentication.

AssumeRole ARN: Enter the Amazon Resource Name (ARN) of the role to assume.

External ID: Enter the External ID to use when assuming the role.

AWS EC2 Authentication

Vault Role: Enter the authentication role to use in Vault.

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

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.

Service Configuration

KMS Key ARN: Enter the Amazon Resource Name (ARN) of the AWS KMS Key to use for encryption. This entry is required.

;

4.6.2. Securing Communications

This page outlines how to protect Cribl Stream Leader to Worker Nodes communications, using an existing 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. Open your SSL certificate file. (The self-signed certificate example used the placeholder name `myCert.pem`.)

2. Copy the file's contents to your clipboard.

3. Paste the content into the Leader's global ⚙ **Settings** (lower left) > **Security** > **Certificates** > **New Certificates** modal > **Certificate** field.

    > You can also skip the preceding two steps, and just upload or drag/drop the file from your filesystem.

4. Open your private key file. (The self-signed certificate example used the placeholder name `myKey.pem`.)

5. Copy the file's contents to your clipboard.

6. Paste the clipboard contents into the same Leader modal's **Private key** field.

    > You can also skip the preceding two steps, and just upload or drag/drop the file from your filesystem.

7. Fill in the **Name** and **Description** fields.

8. If you've uploaded a self-signed certificate, just **Save** it now.

9. If your private key is encrypted, fill in the modal's **Passphrase** with the corresponding key.

10. 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).

Connecting to the Leader Securely

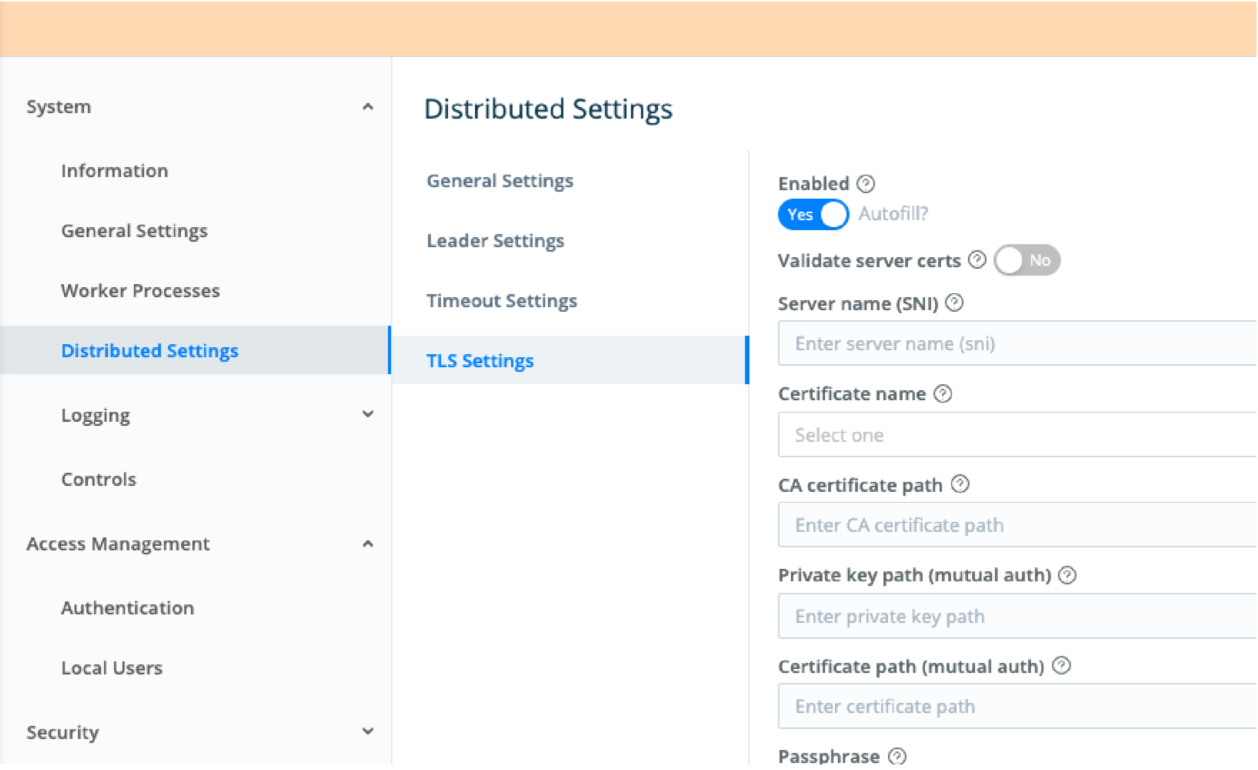
You can configure secure communication between your Leader and Worker Nodes using the UI, the `instance.yml` config file, or environment variables.

Using the UI

Set up secure communication first on the Worker Node, then on the Leader.

Worker Node Setup

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. Select the Worker Node's ⚙ **Settings** (top right) > **System** > **Distributed Settings** > **TLS Settings**.
4. Toggle the **Enabled** slider to `Yes`.
5. Click **Save**.
6. Repeat the preceding steps on each Worker Node.



Enabled: Yes (Autofill?)

Validate server certs: No

Server name (SNI): Enter server name (sni)

Certificate name: Select one

CA certificate path: Enter CA certificate path

Private key path (mutual auth): Enter private key path

Certificate path (mutual auth): Enter certificate path

Passphrase

Leader Setup

Next, return to the Leader's UI:

1. Select global ⚙ **Settings** (bottom left) > **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**.

Using YAML Config File

You can also configure the `$CRIBL_HOME/local/_system/instance.yml` file to ensure that TLS is enabled. Here's the relevant section:

```
distributed:
    tls:
      disabled: false
```

> If you are setting up a Worker Node for the first time using the bootstrapping command, executing that command will automatically update this configuration setting with the appropriate value.

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 tenabled, the Leader checks the certificate against the trust store, to see if it has been signed by a valid certificate authority (CA).

   The Leader checks the list of certificates in the `CA Certificates Path` box first (if populated), then against the list of built-in system certificates.

2. The Leader checks whether the `Common Name` (CN) matches the regular expression in the configuration. Cribl 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.

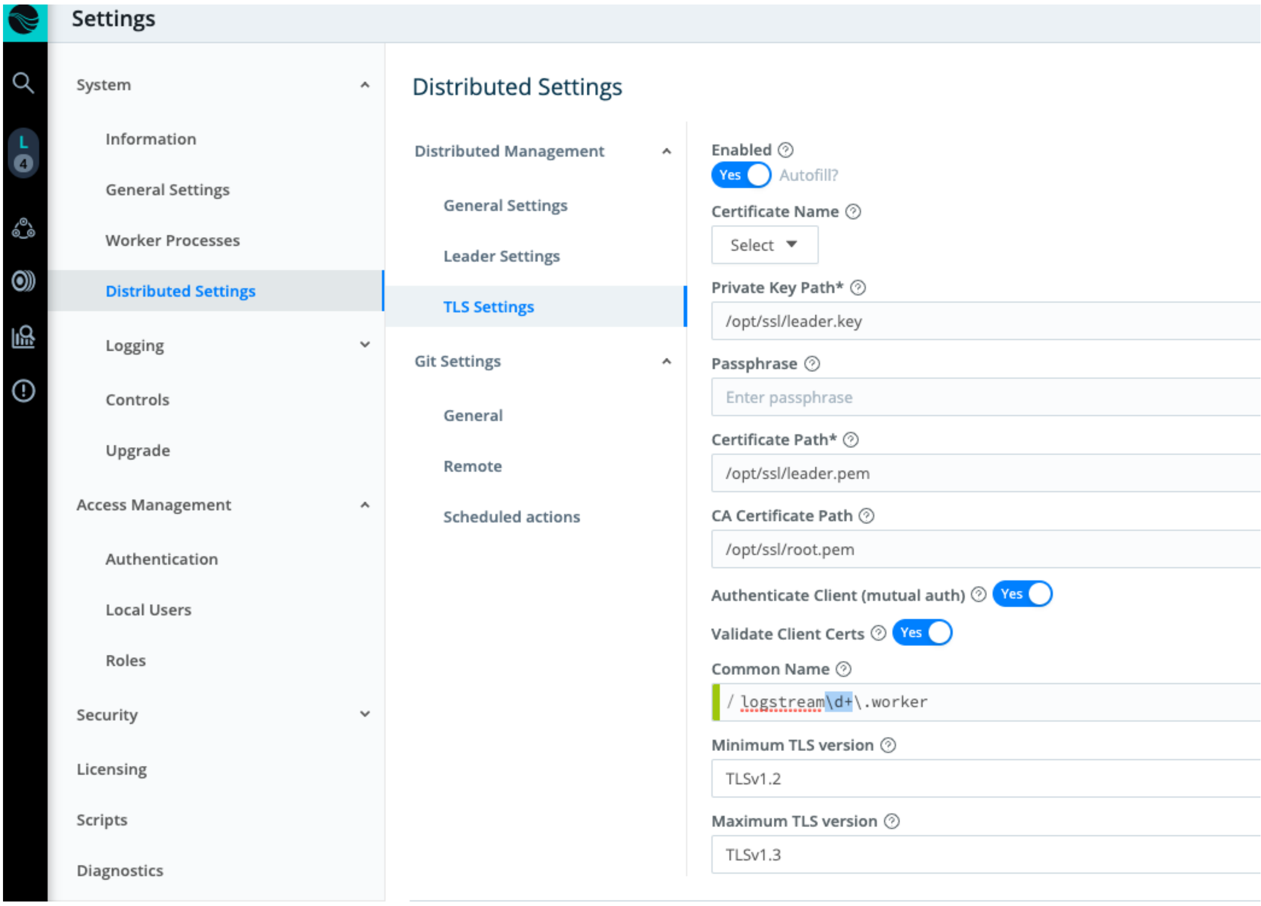
Limitations on TLS Mutual Auth

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

Below is a sample configuration on the Leader:



Enabled: Yes
Autofill?
Certificate Name: Select
Private Key Path*: /opt/ssl/leader.key
Passphrase: Enter passphrase
Certificate Path*: /opt/ssl/leader.pem
CA Certificate Path: /opt/ssl/root.pem
Authenticate Client (mutual auth): Yes
Validate Client Certs: Yes
Common Name: / logstream\d+\.worker
Minimum TLS version: TLSv1.2
Maximum TLS version: TLSv1.3

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 config file, add a few extra items need to the Worker Node's `instance.yml`, as shown here:

```
distributed:
    tls:
      disabled: false
      privKeyPath: /path/to/certs/worker.key
      certPath: /path/to/certs/worker.pem
      caPath: /path/to/certs/root.pem
      requestCert: true
      rejectUnauthorized: true # false if ignoring untrusted certs
```

Using Environment Variables

You can set up TLS mutual authentication by configuring this environment variable, using the format shown in this example:

```
CRIBL_DIST_Master_URL="tls://<authToken>@leader.cribl:4200?
tls.privKeyPath=/path/to/certs/woker.key&tls.certPath=/path/to/certs/wokers.pem&tls.caP
```

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.

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 **+ Add New**.

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.

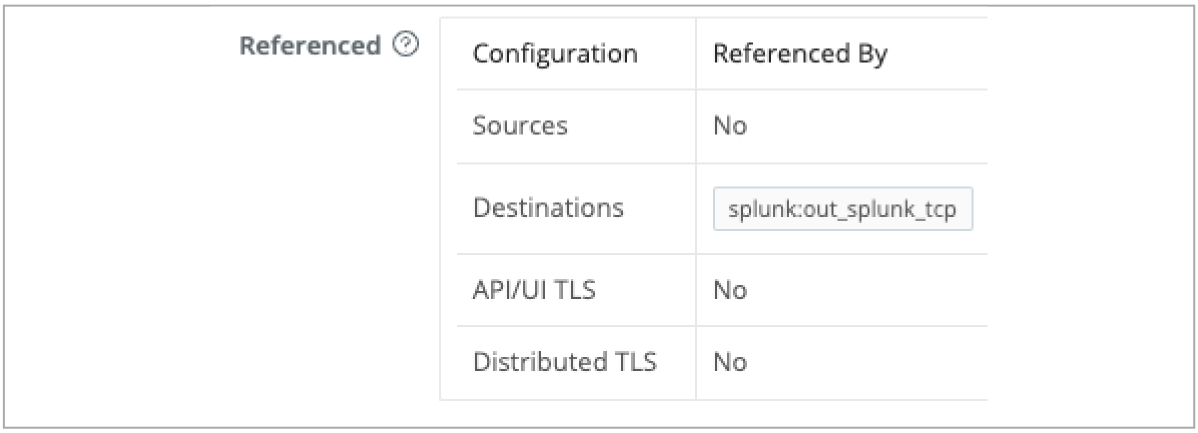
9. Slide **Enabled** on.

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 global ⚙ **Settings** (lower left) > **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)`.

;

4.6.3. 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

;

4.6.4. Encryption

Encryption of Data in Motion

With Cribl Stream, you can encrypt fields or patterns within events in real time, by using `C.Crypto.encrypt()` in a Mask function. The Mask function accepts multiple replacement rules and multiple fields to apply them to.

A **Match regex** defines the pattern of content to be replaced. The **Replace expression** is a JS expression or literal to replace matched content. The `C.Crypto.encrypt()` method can be used here to generate an encrypted string from a value passed to it.

> ### C.Crypto.encrypt() Syntax
>
> (method) `Crypto.encrypt(value: any, keyclass: number, keyId?: string, defaultVal?: string): string` Encrypt the given value with the `keyId`, or with a `keyId` picked up automatically based on `keyclass`.
>
> @param {string | Buffer} `value` – what to encrypt. @param – `keyclass` – if `keyId` isn't specified, pick one at the given `keyclass`. @param – `keyId` - encryption keyId, takes precedence over `keyclass`. @param – `defaultVal` – what to return if encryption fails for any reason; if unspecified, the original value is returned. @returns – if encryption succeeds, the encrypted value; otherwise, `defaultVal` if specified; otherwise, `value`.

Encryption Keys

Symmetric keys can be configured 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>
```

```
keyId  algorithm    keyclass  kms    created          expires     useIV
-----------------------------------------------------------------------
1      aes-256-cbc  0         local  1544906269.316  0           false
2      aes-256-cbc  1         local  1544906272.452  0           false
3      aes-256-cbc  2         local  1544906275.948  1545906275  true
4      aes-256-cbc  3         local  1544906278.026  0           false
```

Adding Keys

Displaying `--help`:

```
$CRIBL_HOME/bin/cribl keys add --help
```

```
Add encryption keys
Usage: [options] [args]

Options:
[-c <keyclass>] - key class to set for the key
[-k <kms>]      - KMS to use, must be configured, see cribl.yml
[-e <expires>]  - expiration time, epoch time
[-i]            - use an initialization vector
 -g <group>     - Group 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
```

```
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:

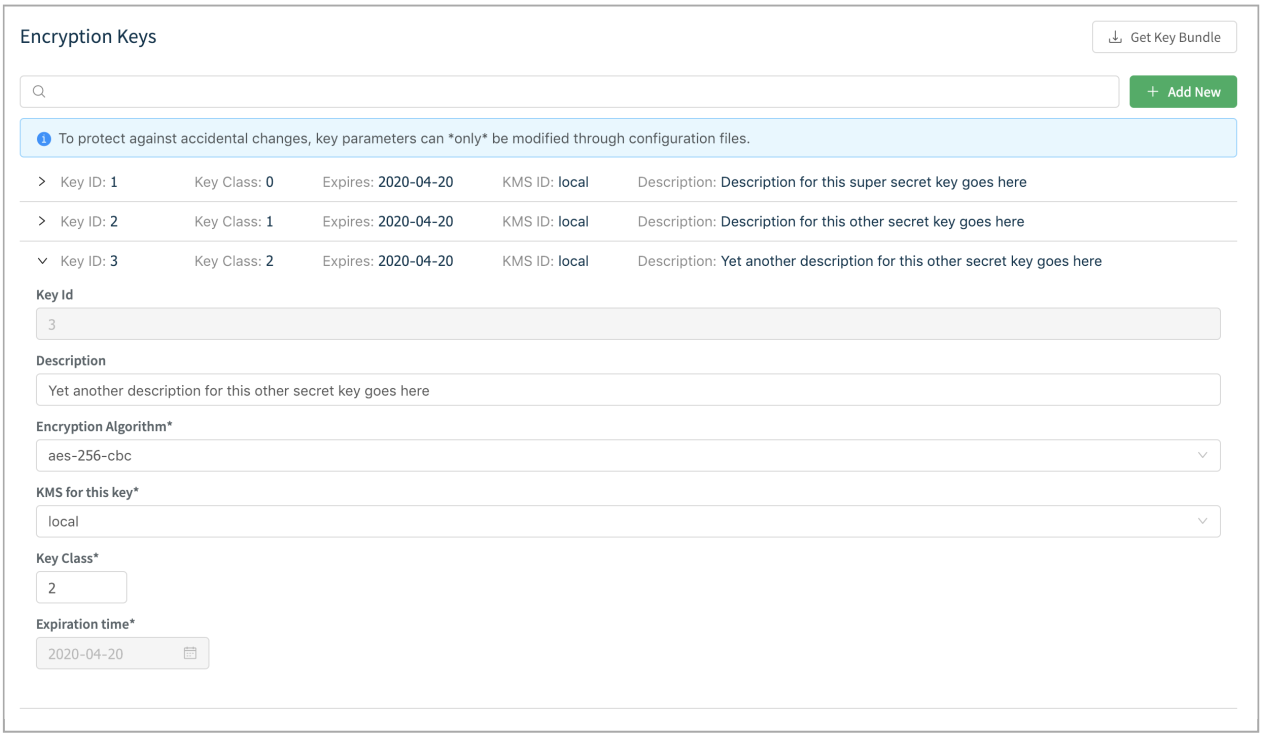
```
$CRIBL_HOME/bin/cribl keys list -g default
```

```
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 global ⚙ **Settings** (lower left) > **Security > Encryption Keys**. In a distributed deployment, for each Group, select **Groups** > `<group-name>` > **Settings** > **Security** > **Encryption Keys**.

Here, you can list and add new keys. To protect against accidental changes, a key's parameters, once saved, can be edited only through configuration files.



Encryption Keys

To protect against accidental changes, key parameters can *only* be modified through configuration files.

> Key ID: 1 Key Class: 0 Expires: 2020-04-20 KMS ID: local Description: Description for this super secret key goes here

> Key ID: 2 Key Class: 1 Expires: 2020-04-20 KMS ID: local Description: Description for this other secret key goes here

Key ID: 3 Key Class: 2 Expires: 2020-04-20 KMS ID: local Description: Yet another description for this other secret key goes here

Key Id
3

Description
Yet another description for this other secret key goes here

Encryption Algorithm*
aes-256-cbc

KMS for this key*
local

Key Class*
2

Expiration time*
2020-04-20

List or create keys through Cribl Stream's UI

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/`.

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).

;

4.6.5. 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 --spunk-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.

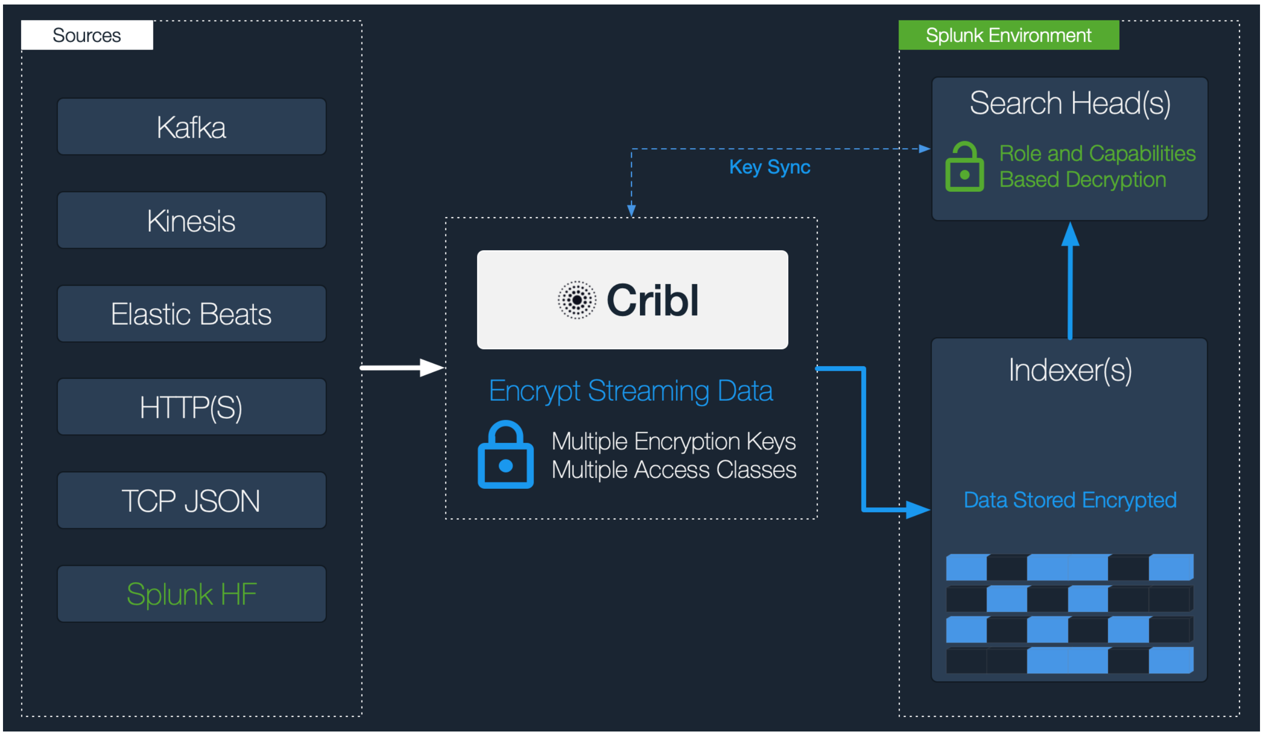
Restricting Access with Splunk Capabilities

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`.

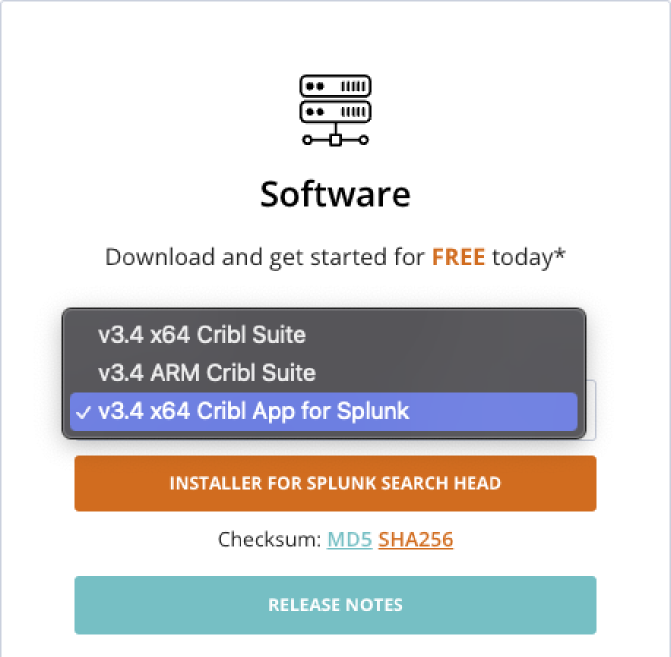
CAPABILITY NAME	CORRESPONDING CRIBL KEY CLASS
`cribl_keyclass_1`	`1`
`cribl_keyclass_2`	`2`
…	…
`cribl_keyclass_N`	`N`

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 `$SPLUNK_HOME/etc/apps` directory.
   As of LogStream v1.7, the app will run in search head mode by default. If the app has previously been installed and later modified, you can convert it to search head mode with the command: `$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).

;

4.7. 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 (Beta)
- Health Endpoint

Monitoring Page

Select **Monitoring** from the left nav in distributed deployments (Edge, Stream) or top nav in single-instance deployments (Edge, Stream). 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/Fleets and all Workers/Edge Nodes. You can use the drop-downs at the upper right to isolate individual Groups/Fleets, or individual Workers/Edge Nodes.

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** 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.

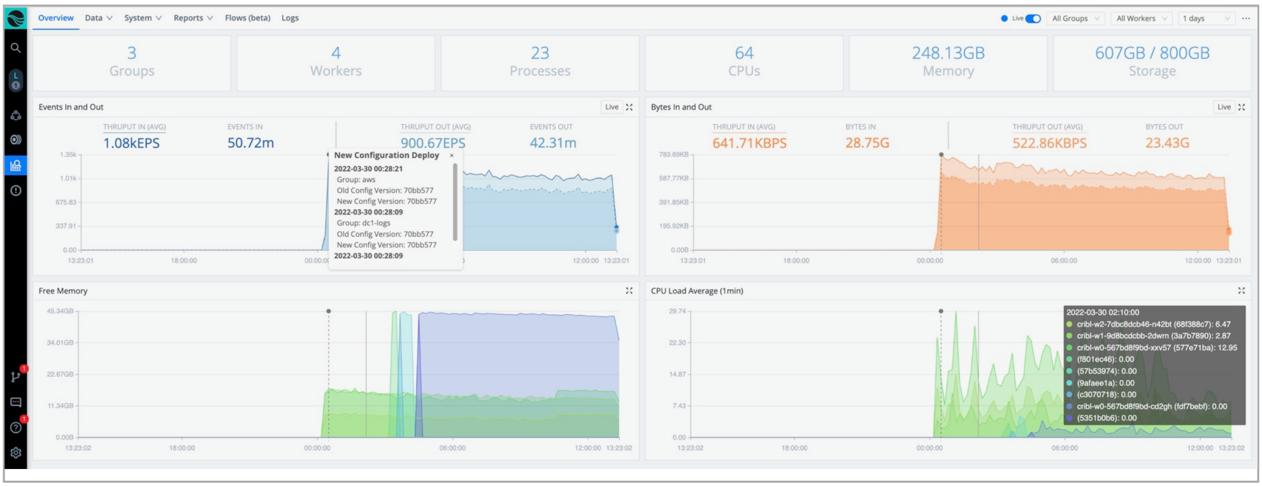
Byte-related charts show the uncompressed amounts and rates of data processed over the selected time range:

- Events (total) in and out
- Events per second in and out
- Bytes (total) in and out
- Bytes per second in and out

We measure bytes in and out based on the size of `_raw`, if this field is present and is of type string. Otherwise, we use the size of read (uncompressed) data.

The displayed **CPU Load Average** is an average per Worker/Edge Node Process, updated at 1-minute granularity. (It is not an average for the Worker/Edge 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.



New Configuration Deploy ×
2022-03-30 00:28:21
Group: aws
Old Config Version: 70bb577
New Config Version: 70bb577
2022-03-30 00:28:09
Group: dc1-logs
Old Config Version: 70bb577
New Config Version: 70bb577
2022-03-30 00:28:09

2022-03-30 02:10:00
● cribl-w2-7dbc8dcb46-n42bt (68f388c7): 6.47
● cribl-w1-9d8bcdcbb-2dwrn (3a7b7890): 2.87
● cribl-w0-567bd8f9bd-xxv57 (577e71ba): 12.95
● (f801ec46): 0.00
● (57b53974): 0.00
● (9afaee1a): 0.00
● (c3070718): 0.00
● cribl-w0-567bd8f9bd-cd2gh (fdf7bebf): 0.00
● (5351b0b6): 0.00

Monitoring page

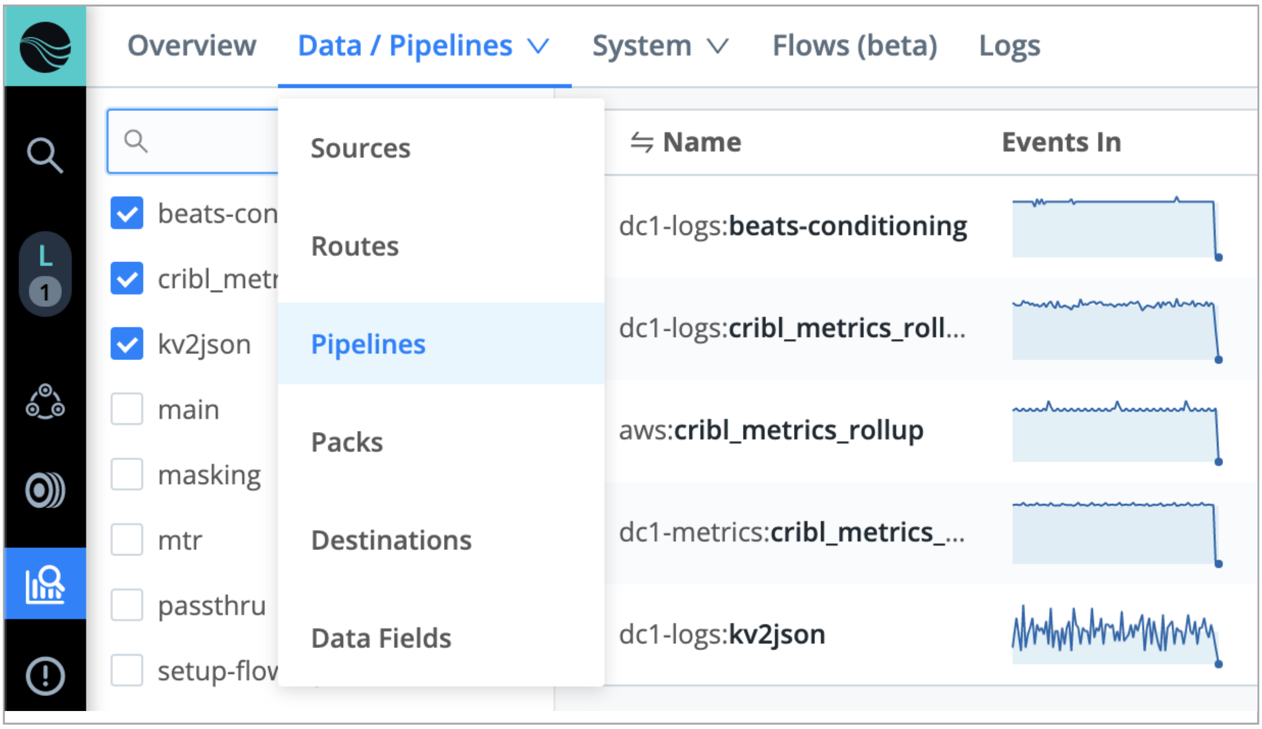
> Except for these configuration change markers, Monitoring data does not persist across Cribl Stream restarts. Keep this in mind before you restart the server.

Data Monitoring

From the **Monitoring** page's top nav, open the **Data** submenu to isolate throughput for any of the following:

- Sources
- Routes
- Pipelines
- Packs
- Destinations

- Data Fields

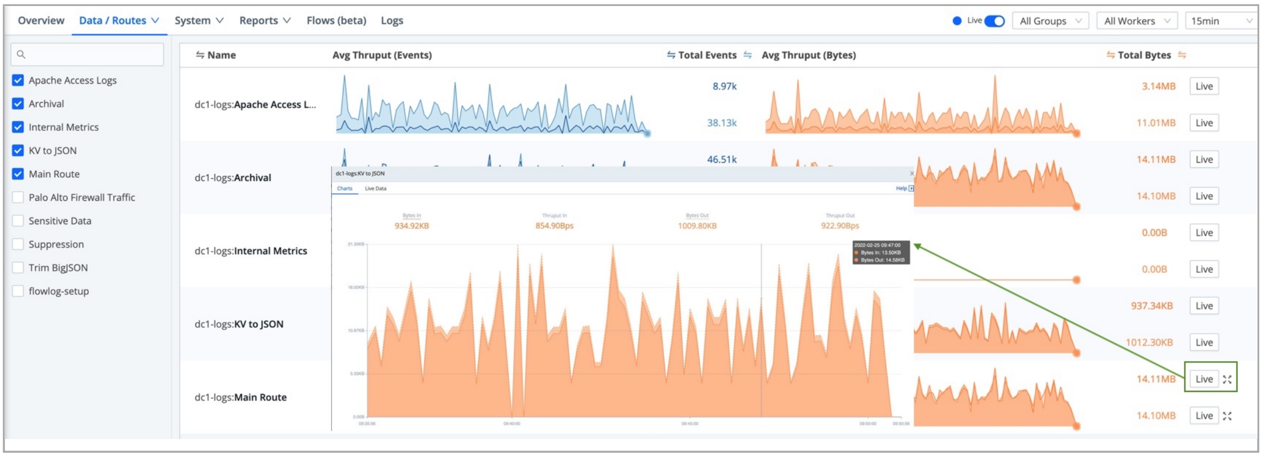


☑ beats-con
☑ cribl_metr
☑ kv2json
☐ main
☐ masking
☐ mtr
☐ passthru
☐ setup-flov

⇋ Name	Events In
dc1-logs:**beats-conditioning**	
dc1-logs:**cribl_metrics_roll...**	
aws:**cribl_metrics_rollup**	
dc1-metrics:**cribl_metrics_...**	
dc1-logs:**kv2json**	

Monitoring > Data submenu (Pipelines selected)

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.

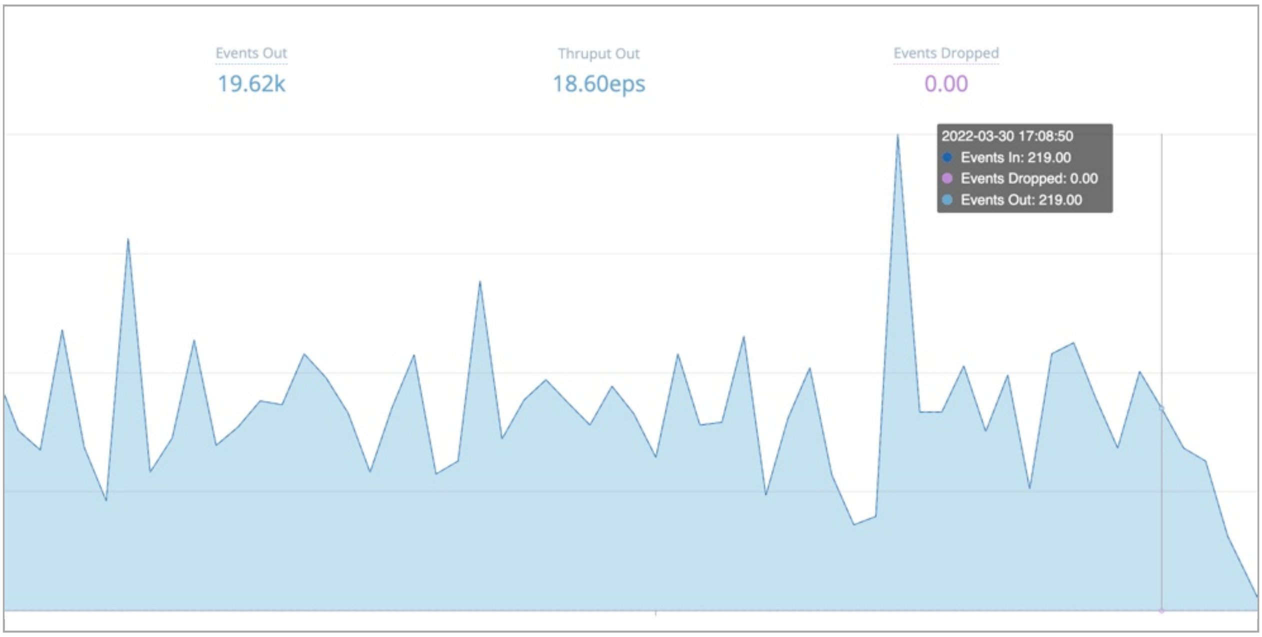


☑ Apache Access Logs
☑ Archival
☑ Internal Metrics
☑ KV to JSON
☑ Main Route
☐ Palo Alto Firewall Traffic
☐ Sensitive Data
☐ Suppression
☐ Trim BigJSON
☐ flowlog-setup

⇋ Name	Avg Thruput (Events)	⇋ Total Events ⇋	Avg Thruput (Bytes)	⇋ Total Bytes ⇋	
dc1-logs:**Apache Access L...**		8.97k 38.13k		3.14MB 11.01MB	Live Live
dc1-logs:**Archival**		46.51k		14.11MB 14.10MB	Live Live
dc1-logs:**Internal Metrics**				0.00B 0.00B	Live Live
dc1-logs:**KV to JSON**				937.34KB 1012.30KB	Live Live
dc1-logs:**Main Route**				14.11MB 14.10MB	Live Live

Sparklines and fly-out

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

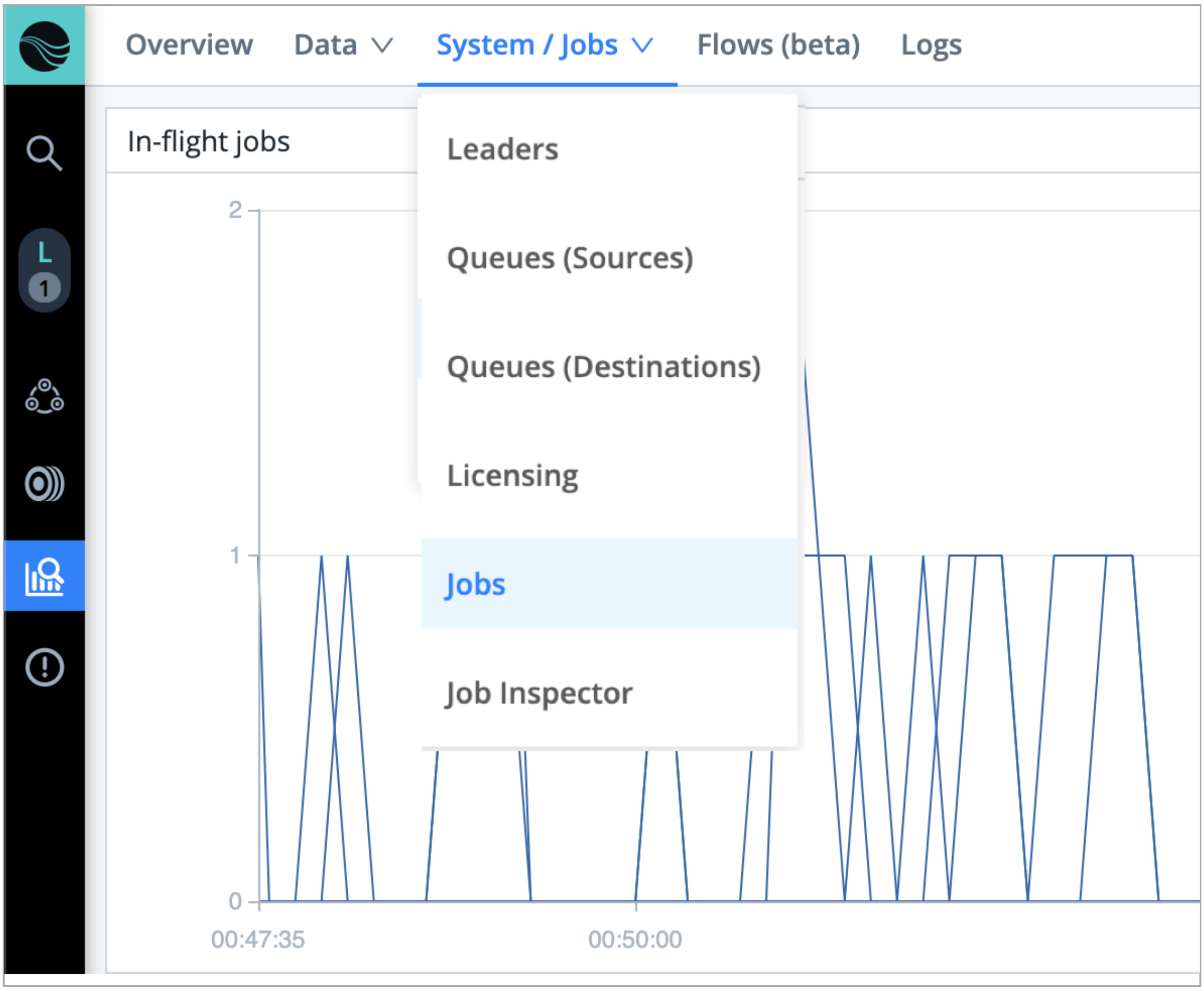


Throughput details

System Monitoring

From the **Monitoring** page's top nav, open the **System** submenu to isolate throughput for any of the following:

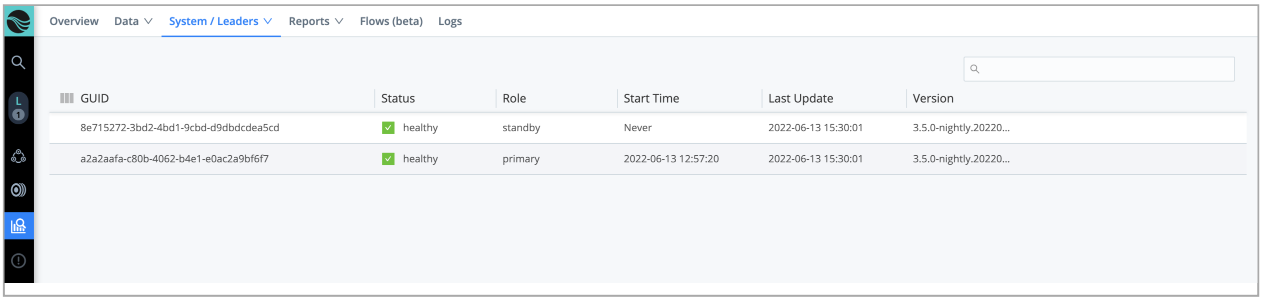
- Leaders (see Second Leader)
- Queues (see Persistent Queues)
- Licensing
- Jobs (and tasks in-flight, see Collector Sources)
- Job Inspector



Monitoring > System submenu (Jobs in-flight selected)

Leaders

Select **Monitoring** (side or top nav) > **System** > **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).



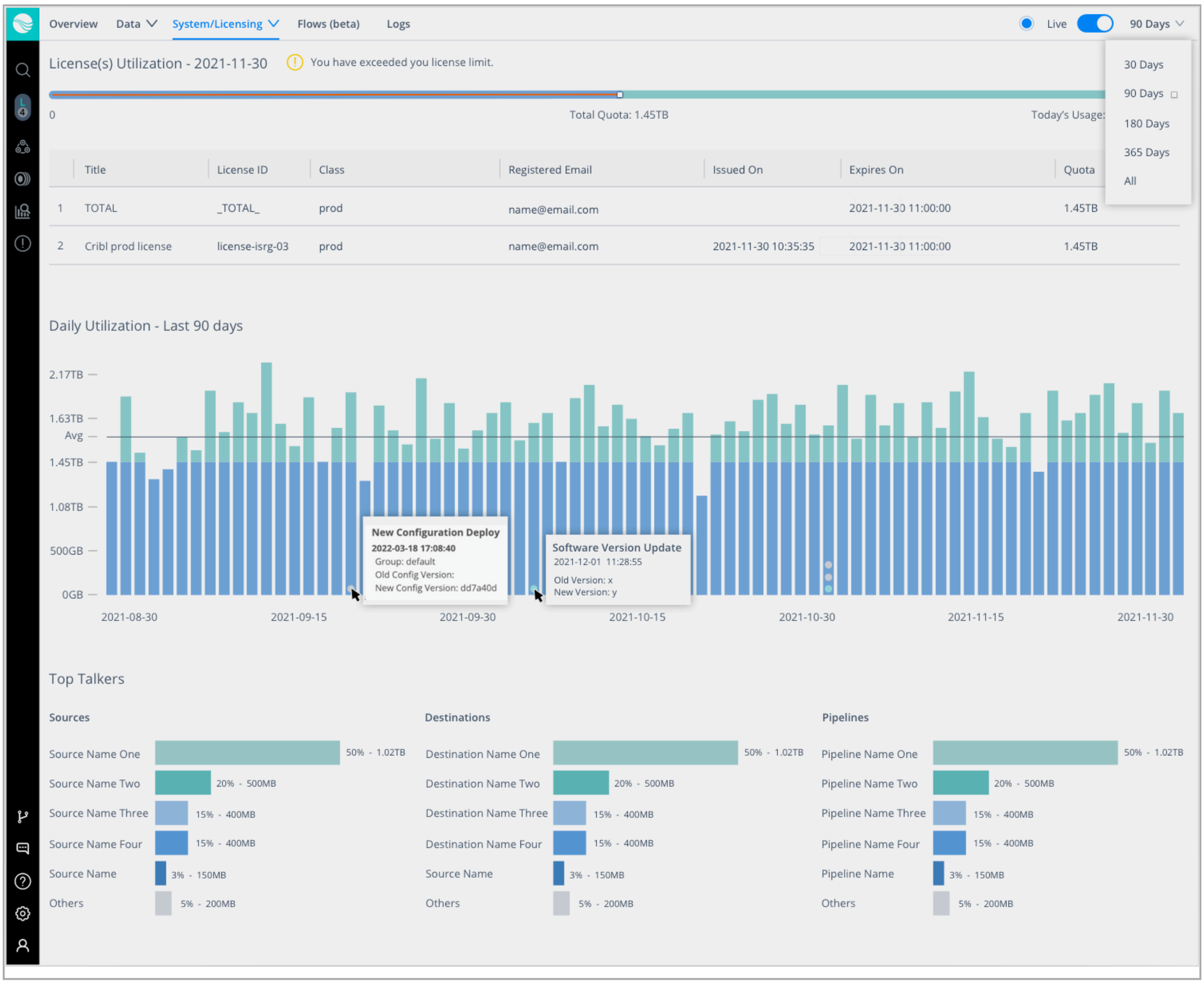
GUID	Status	Role	Start Time	Last Update	Version
8e715272-3bd2-4bd1-9cbd-d9dbdcdea5cd	✓ healthy	standby	Never	2022-06-13 15:30:01	3.5.0-nightly.20220...
a2a2aafa-c80b-4062-b4e1-e0ac2a9bf6f7	✓ healthy	primary	2022-06-13 12:57:20	2022-06-13 15:30:01	3.5.0-nightly.20220...

Monitoring Leader Nodes

Licensing

Select **System** > **Licensing** from the Monitoring page's top nav to check your licenses' expiration dates, and daily data throughput quotas. You can also compare your daily data throughput against your license quota – and against granular and average throughput over the last 30 to 365 days. Highlights include:

- A horizontal bar indicates license usage against your quota.
- Tooltips display details about data usage, data amount over/under license quota, and data percentage over/under license quota.
- Dots on the daily usage bar graph represent configuration changes in the system.



License(s) Utilization - 2021-11-30

You have exceeded you license limit.

	Title	License ID	Class	Registered Email	Issued On	Expires On	Quota
1	TOTAL	_TOTAL_	prod	name@email.com		2021-11-30 11:00:00	1.45TB
2	Cribl prod license	license-isrg-03	prod	name@email.com	2021-11-30 10:35:35	2021-11-30 11:00:00	1.45TB

Daily Utilization - Last 90 days

New Configuration Deploy
2022-03-18 17:08:40
Group: default
Old Config Version:
New Config Version: dd7a40d

Software Version Update
2021-12-01 11:28:55
Old Version: x
New Version: y

Top Talkers

Sources	
Source Name One	50% - 1.02TB
Source Name Two	20% - 500MB
Source Name Three	15% - 400MB
Source Name Four	15% - 400MB
Source Name	3% - 150MB
Others	5% - 200MB

Destinations	
Destination Name One	50% - 1.02TB
Destination Name Two	20% - 500MB
Destination Name Three	15% - 400MB
Destination Name Four	15% - 400MB
Source Name	3% - 150MB
Others	5% - 200MB

Pipelines	
Pipeline Name One	50% - 1.02TB
Pipeline Name Two	20% - 500MB
Pipeline Name Three	15% - 400MB
Pipeline Name Four	15% - 400MB
Pipeline Name	3% - 150MB
Others	5% - 200MB

Monitoring > Licensing

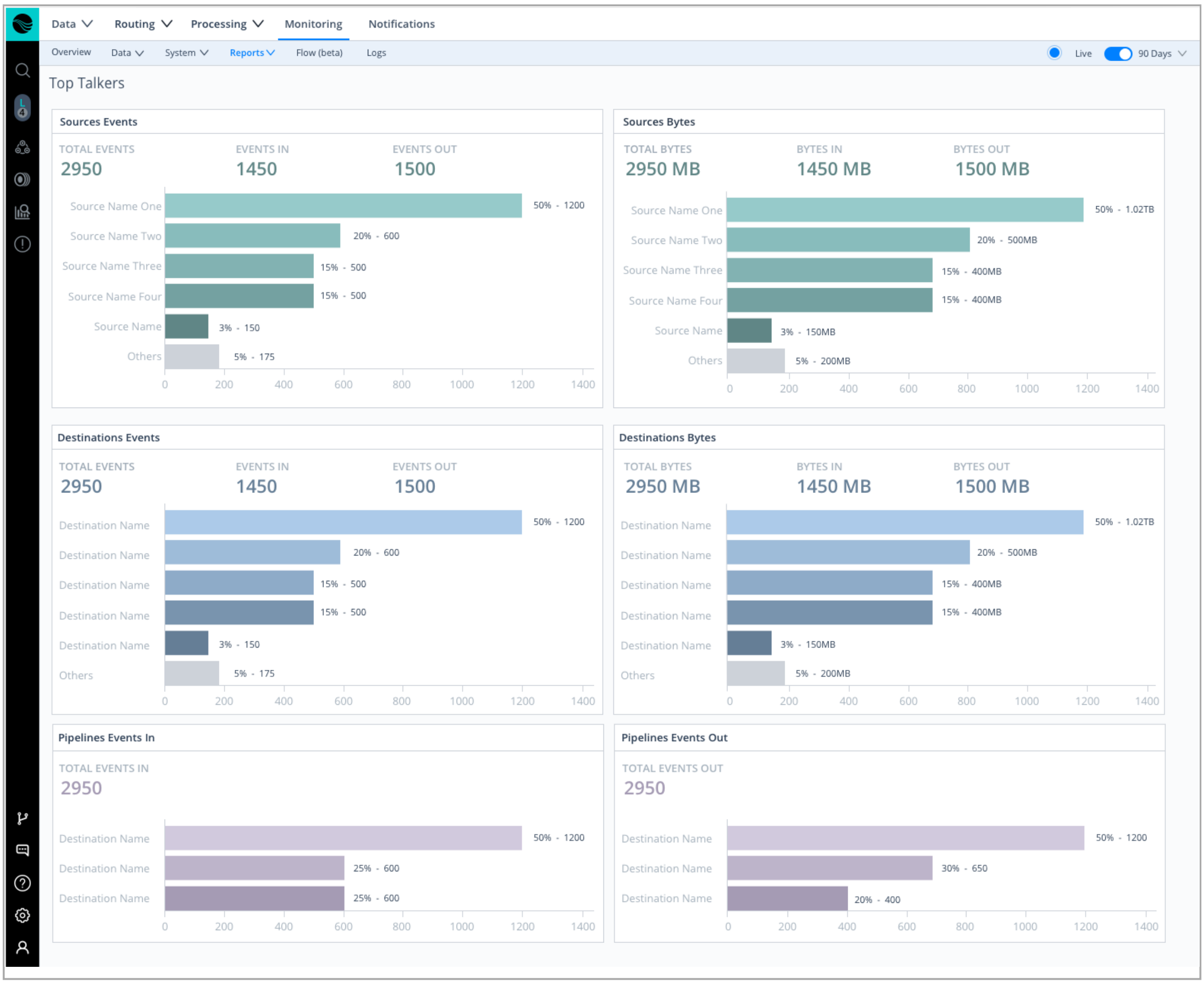
> Even on single-instance deployments, you must have `git` installed in order for the Monitoring > **Licensing** page to display configuration change markers.

Job Inspector

Select **System** > **Job Inspector** from the Monitoring page's top nav 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.

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 (Beta)

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/Edge Node Process(es) Logs – These logs are emitted by all the Worker/Edge Node Processes, and are very common on single-instance deployments and Worker/Edge Nodes. 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`

> For details about generated log files, see Internal Logs. To work with logs in Cribl Stream's UI, see Search Internal Logs.

Cribl Stream rotates logs every 5 MB, keeping the most recent 5 logs. In a distributed deployment (Edge, Stream), all Workers/Edge Nodes forward their metrics to the Leader Node, which then consolidates them to provide a deployment-wide view.

Forward Logs and Metrics Externally

Cribl Stream supports forwarding internal logs and metrics to your preferred external monitoring solution. To make internal data available to send out, go to **Sources** and enable the Cribl Internal Source.

This will send internal logs and metrics down through Routes and Pipelines, just like another data source. Both logs and metrics will have a field called `source`, set to the value `cribl`, which you can use in Routes' filters.

Note that the only logs supported here are Worker/Edge Node Process logs (see Types of Logs above). You can, however, use a Script Collector to listen for API Server or Worker/Edge Node Group/Fleet events.

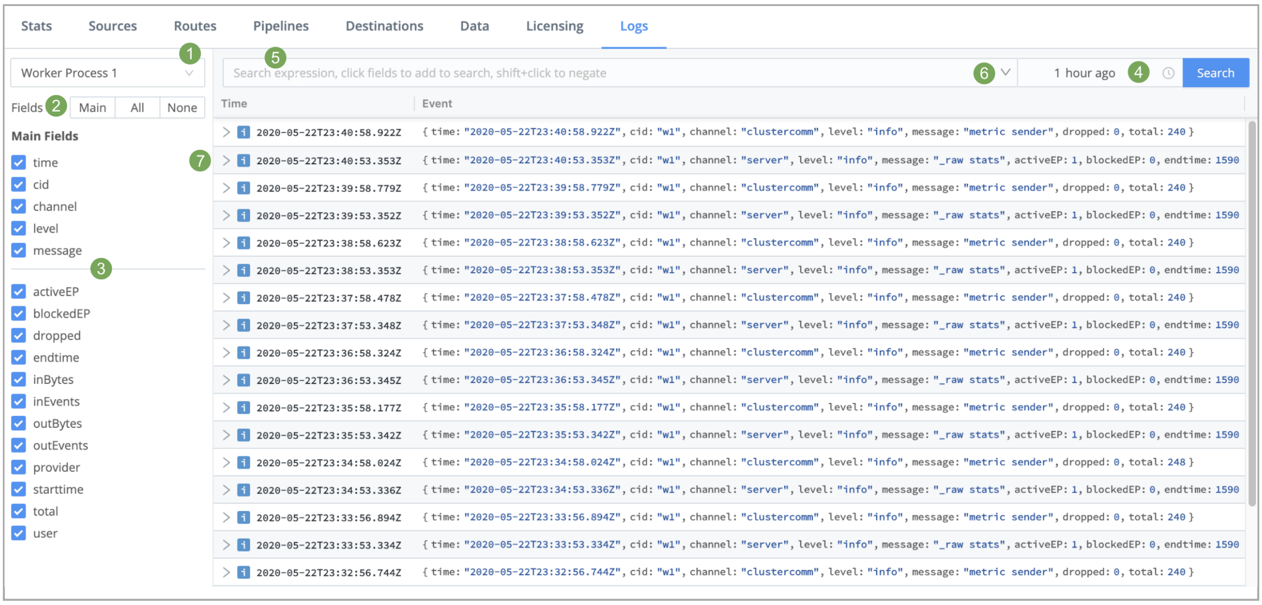
For recommendations about useful Cribl metrics to monitor, see Internal Metrics.

> ### CriblMetrics Override
>
> The **Disable field metrics** setting – in global ⚙ **Settings** (lower left) > **System > General Settings > Limits -** applies only to metrics sent to the Leader Node. When the **Cribl Internal** Source is enabled, Cribl Stream ignores this **Disable field metrics** setting, and full-fidelity data will flow down the Routes.

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/Edge Node. This enables you to query across all internal logs for strings of interest.

The labels on this screenshot highlight the key controls you can use (see the descriptions below):



☑ time
☑ cid
☑ channel
☑ level
☑ message

☑ activeEP
☑ blockedEP
☑ dropped
☑ endtime
☑ inBytes
☑ inEvents
☑ outBytes
☑ outEvents
☑ provider
☑ starttime
☑ total
☑ user

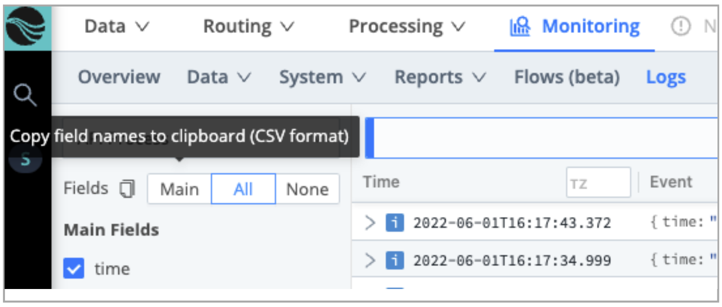
```
2020-05-22T23:40:58.922Z  { time: "2020-05-22T23:40:58.922Z", cid: "w1", channel: "clustercomm", level: "info", message: "metric sender", dropped: 0, total: 240 }
2020-05-22T23:40:53.353Z  { time: "2020-05-22T23:40:53.353Z", cid: "w1", channel: "server", level: "info", message: "_raw stats", activeEP: 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 (Edge, Stream), this list will be hierarchical, with Workers/Edge Nodes 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.



```
2022-06-01T16:17:43.372   { time: "
2022-06-01T16:17:34.999   { time: "
```

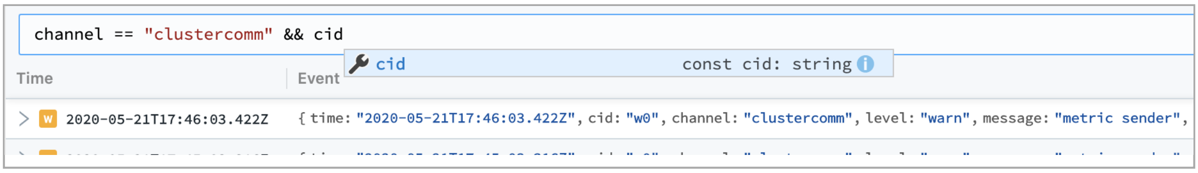
Monitoring - Copy Fields Icon

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:

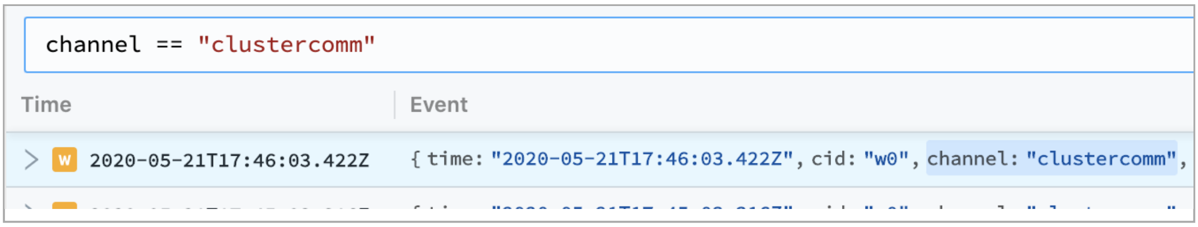


```
channel == "clustercomm" && cid
cid                                   const cid: string
```

```
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:

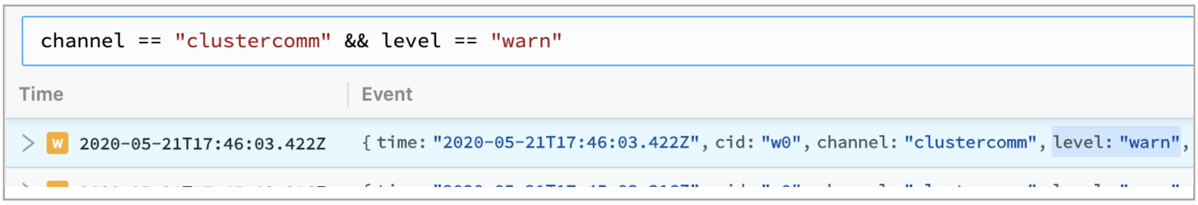


```
channel == "clustercomm"
```

```
2020-05-21T17:46:03.422Z   { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm",
```

Click to add a field

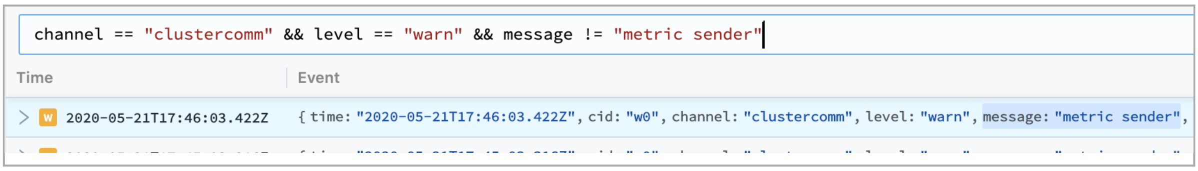
   Click other fields to append them to a query:



```
channel == "clustercomm" && level == "warn"
Time                        Event
> W 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:

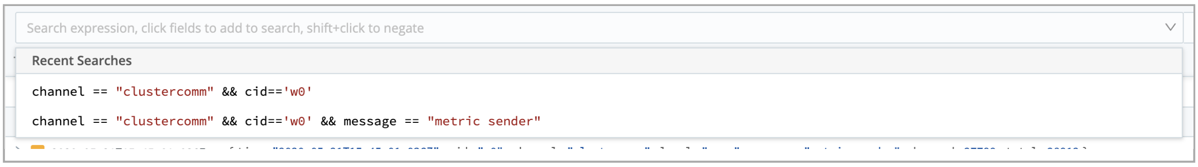


```
channel == "clustercomm" && level == "warn" && message != "metric sender"
Time                        Event
> W 2020-05-21T17:46:03.422Z  { time: "2020-05-21T17:46:03.422Z", cid: "w0", channel: "clustercomm", level: "warn", message: "metric sender",
```

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:

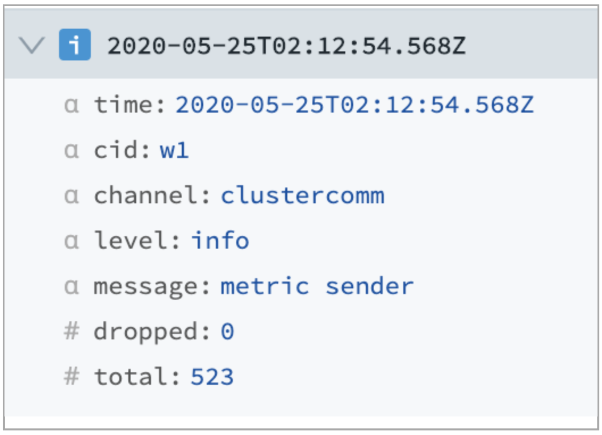


```
Search expression, click fields to add to search, shift+click to negate
Recent Searches
channel == "clustercomm" && cid=='w0'
channel == "clustercomm" && cid=='w0' && message == "metric sender"
```

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:



```
∨ i 2020-05-25T02:12:54.568Z
    a time: 2020-05-25T02:12:54.568Z
    a cid: w1
    a channel: clustercomm
    a level: info
    a message: metric sender
    # dropped: 0
    # total: 523
```

Log event details

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//Edge Node Fleet.

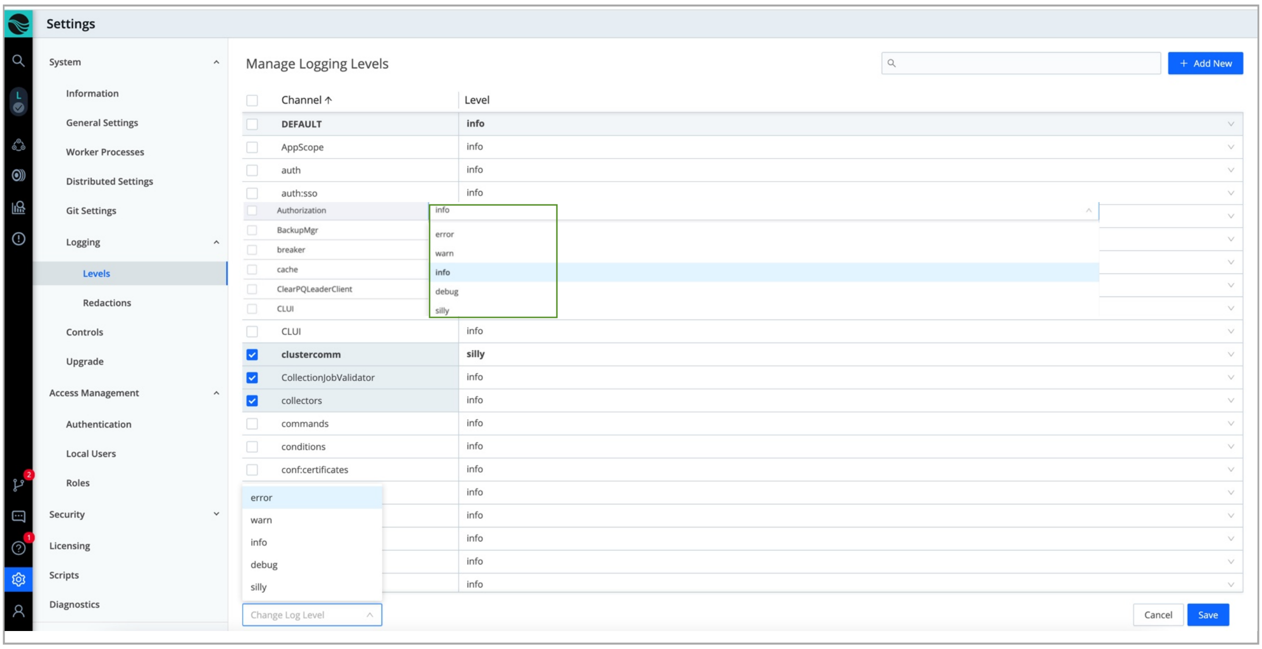
Change Logging Levels

To adjust logging levels:

- In a single-instance deployment, or for the Leader Node's own logs, select global ⚙ **Settings** (lower left) > **System > Logging > Levels**.
- In a distributed deployment's left nav, first select **Groups**, then click into the group you want to configure. Next, select group **Settings** (upper right) > **System > Logging > Levels**.

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

- Modify one channel by clicking its **Level** column. In the resulting drop-down, you can set a verbosity level ranging from **error** up to **debug**. (Top of composite screenshot below.)
- Modify multiple channels by selecting their check boxes, then clicking the **Change log level** drop-down at the bottom of the page. (Bottom of composite screenshot below.) You can select all channels at once by clicking the top check box. You can search for channels at top right.



Manage Logging Levels

□	Channel ↑	Level
□	DEFAULT	info
□	AppScope	info
□	auth	info
□	auth:sso	info
□	Authorization	info
□	BackupMgr	
□	breaker	
□	cache	
□	ClearPQLeaderClient	
□	CLUI	
□	CLUI	info
☑	clustercomm	silly
☑	CollectionJobValidator	info
☑	collectors	info
□	commands	info
□	conditions	info
□	conf:certificates	info
		info
		info
		info
		info
		info

Manage Logging Levels page

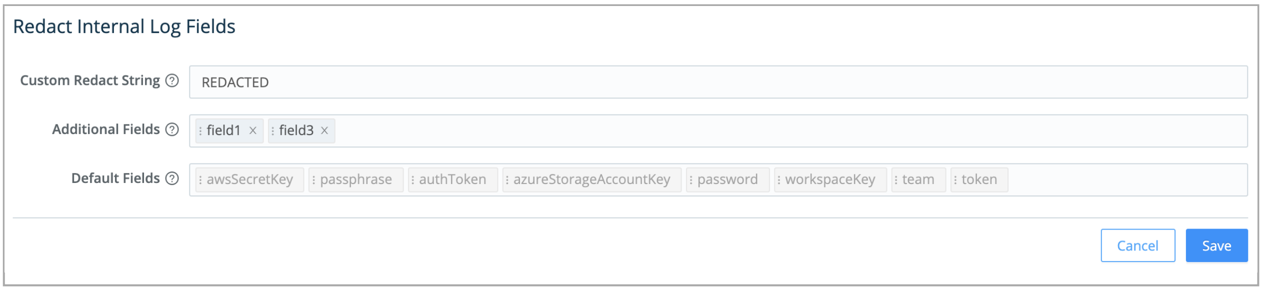
> The `silly` (ultra-verbose) logging level is available for all channels. However, it provides additional metrics information only for inputs.

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, or for the Leader Node's own logs, select global ⚙ **Settings** (lower left) > **System > Logging > Redactions**.

- In a distributed deployment's left nav, first select **Groups**, then click into the group you want to configure. Next, select that group's **Settings** (upper right) > **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`).

;

4.7.1. 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 StreamUI (in the same way that it reports internal logs at **Monitoring** > **Logs**). To expose metrics for capture or routing, enable the **Cribl Internal** Source > **CriblMetrics** section.

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.

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 documented 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": { "ci": "http:http",   "input": "http:http" }}, "val": 0},
    { "model": { "ci": "cribl:CriblLogs",   "input": "cribl:CriblLogs" }}, "val":
0},
    { "model": { "ci": "cribl:CriblMetrics",   "input": "cribl:CriblMetrics" }},
"val": 0},
    { "model": { "ci": "datagen:DatagenWeblog",   "input": "datagen:DatagenWeblog"
}}, "val": 0 }
    ],
  "health.outputs": [
    { "model": { "output": "devnull:devnull" }}, "val": 0},
    { "model": { "output": "router:MyOut1" }}, "val": 0},
    { "model": { "output": "tcpjson:MyTcpOut1" }}, "val": 0},
    { "model": { "output": "router:MyOut2" }}, "val": 0},
    { "model": { "output": "tcpjson:MyTcpOut2" }}, "val": 0},
    { "model": { "output": "router:MyOut3" }}, "val": 0},
    { "model": { "output": "router:MyOut4" }}, "val": 0 }
    ],
```

Worker/Edge Node Resource Metrics

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

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

Persistent Queue Metrics

These metrics are valuable for monitoring Persistent Queues' behavior:

- `pq.queue_size` – Total bytes in the queue.
- `pq.in_bytes` – Total bytes in the queue for the given **Time Window**.
- `pq.in_events` – Number of events in the queue for the given **Time Window**.
- `pq.out_bytes` – Total bytes flushed from the queue for the given **Time Window**.
- `pq.out_events` – Number of events flushed from the queue for the given **Time Window**.

These are aggregate metrics. But you can distinguish unique metrics per queue Destination, using the `#output=<id>` dimension. For example, `pq.out_events|#output=kafka` would be one unique metric; `pq.out_events|#output=camus` would be another.

Other Internal Metrics

The Cribl Internal Source emits other metrics that, when displayed in downstream dashboards, can be useful for understanding Cribl 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, per Destination, per Worker/Edge NodeProcess.
- `cribl.logstream.host.in_bytes` – Inbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.in_events` – Inbound events from a given host (host is a characteristic of the data).
- `cribl.logstream.host.out_bytes` – Outbound bytes from a given host (host is a characteristic of the data).
- `cribl.logstream.host.out_events` – Outbound events from a given host (host is a characteristic of the data).
- `cribl.logstream.route.in_bytes` – Inbound bytes per Route.
- `cribl.logstream.route.in_events` – Inbound events per Route.
- `cribl.logstream.route.out_bytes` – Outbound bytes per Route.
- `cribl.logstream.route.out_events` – Outbound events per Route.
- `cribl.logstream.sourcetype.in_bytes` – Inbound bytes per sourcetype.
- `cribl.logstream.sourcetype.in_events` – Inbound events per sourcetype.
- `cribl.logstream.sourcetype.out_bytes` – Outbound bytes per sourcetype.
- `cribl.logstream.sourcetype.out_events` – Outbound events per sourcetype.

Dimensions

Cribl 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.

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 global ⚙ **Settings** (lower left) > **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.

;

4.7.2. Internal Logs

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

Leader Node Logs (Distributed)

The `API/main` process emits the following logs into the Leader Node's `$CRIBL_HOME/log/` directory:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Principal log in Cribl 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 Config Helper process for each Worker Group/Fleet emits the following log in `$CRIBL_HOME/log/group/GROUPNAME`:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about config maintenance, previews, etc.	`GROUPNAME > Config helper`

Worker/Edge Node Logs (Distributed)

The API Process emits the following log in `$CRIBL_HOME/log/`:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about the Worker/Edge Node communicating with the Leader Node (i.e., with its API Process) and other API requests, e.g., sending metrics, reaping job artifacts.	`GROUPNAME > Worker: HOSTNAME > API Process`

Each Worker Process emits the following log in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about the Worker/Edge Node processing data.	`GROUPNAME > Worker:HOSTNAME > Worker Process N`

For convenience, the UI aggregates the Worker/Edge Node Process logs as follows:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
N/A	Aggregation of all the `Worker Process N` logs and the API Process log.	`GROUPNAME > WORKER_NAME`

Single-Instance Logs

The `API/main` process emits the same logs that it does for a distributed deployment, in `$CRIBL_HOME/log/`:

- `cribl.log`
- `access.log`
- `audit.log`
- `notifications.log`
- `ui-access.log`

Each Worker/Edge Node Process emits the following log in `$CRIBL_HOME/log/worker/N/`, where `N` is the Worker/Edge Node Process ID:

LOGFILE NAME	DESCRIPTION	EQUIVALENT ON LOGS PAGE
`cribl.log`	Messages about the Worker/Edge Node processing data.	`GROUPNAME > Worker:HOSTNAME > Worker Process N`

`_raw stats` Event Fields

Each Worker/Edge Node Process logs this information at a 1-minute frequency. So each event's scope covers only that Worker/Edge Node Process, over a 1-minute time span defined by the `startTime` and `endTime` fields.

Sample Event

```
{"time":"2021-11-24T15:12:05.713Z","cid":"w1","channel":"server","level":"info",
"message":"_raw stats","inEvents":31237,"outEvents":44791,"inBytes":7820263,
"outBytes":14701001,"starttime":1637766660,"endtime":1637766720,"activeCxn":0,"openCxn":
136,"closeCxn":135,"pqInEvents":0,"pqOutEvents":0,"pqInBytes":0,"pqOutBytes":0,"pqTotalB
ytes":0,"droppedEvents":1113,"activeEP":41,"blockedEP":5,"cpuPerc":23.34,"mem":
{"heap":119,"ext":79,"rss":380}}
```

Field Descriptions

FIELD	DESCRIPTION
`inEvents`	Number of events received from all inputs after Event Breakers are applied. This can be larger than `outEvents` if events are dropped via Drop, Aggregation, Suppression, Sampling, or similar Functions.
`outEvents`	Number of events sent out to all Destinations. This can be larger than `inEvents` due to creating event clones or entirely new unique events. (E.g., when using the Aggregation Function.)
`inBytes`	Number of bytes received from all Sources (based only off `_raw`).
`outBytes`	Number of bytes sent to all Destinations (based only off `_raw`).
`startTime`	The beginning of the timespan represented by these metrics.
`endTime`	The end of the timespan represented by these metrics. (Will always be 60 seconds after `startTime`.)
`activeCxn`	Number of TCP connections newly opened at the time the `_raw` stats are logged. (This is a gauge when exported in internal metrics, and can otherwise be ignored as an instantaneous measurement. Only some application protocols count toward this; e.g., any HTTP-based Source does not count.)
`openCxn`	Same as `activeCxn`, but tracked as a counter rather than a gauge. So `openCxn` will show all connections newly opened each minute, and is more accurate than using `activeCxn`.

FIELD	DESCRIPTION
`closeCxn`	Number of TCP connections that were closed.
`pqInEvents`	Number of events that were written to Persistent Queues, across all Destinations.
`pqOutEvents`	Number of events that were flushed from Persistent Queues, across all Destinations.
`pqInBytes`	Number of bytes that were written to Persistent Queues, across all Destinations.
`pqOutBytes`	Number of bytes that were flushed from Persistent Queues, across all Destinations.
`pqTotalBytes`	Amount of data currently stored in Persistent Queues, across all Destinations.
`droppedEvents`	This is equivalent to the `total.dropped_events` metric. Drops can occur from Pipeline Functions, from Destination Backpressure, or from other errors. Any event not sent to a Destination is considered dropped.
`activeEP`	Number of currently active event processors (EPs). EPs are used to process events through Breakers and Pipelines as the events are received from Sources and sent to destinations. EPs are typically created per TCP connection (such as for HTTP).
`blockedEP`	Number of currently blocked event processors (caused by blocking Destinations).
`cpuPerc`	CPU utilization from the combined user and system activity over the last 60 seconds.
`mem.heap`	Heap section of process memory, in bytes.
`mem.ext`	External section of process memory, in bytes.
`mem.rss`	Resident set size section of process memory, in bytes.

;

4.8. Notifications

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

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

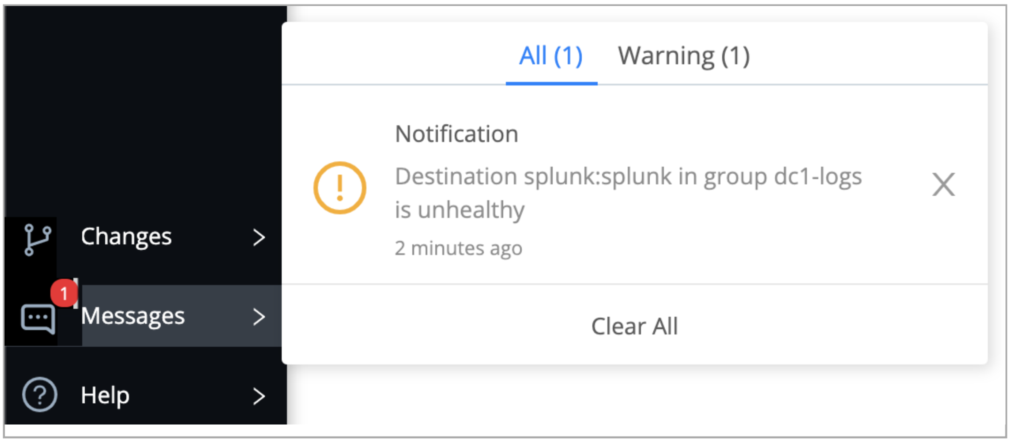
Notifications are not designed to take the place of alerts on your overall infrastructure's health – but they warn you about conditions that could impede expected data flow into and out of Cribl Stream.

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

Notifications and Targets

Every Notification is sent to one or more **targets**. By default, any Notification that you configure will have a target of `System Messages`. This means that when a Notification is triggered, it will add an indicator on the left nav's 💬 **Messages** tab. Click this to view details in a fly-out, as shown below.

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



Notification

Destination splunk:splunk in group dc1-logs is unhealthy

2 minutes ago

System Messages pane

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

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, and can create and manage Notifications and targets

Configuring Notifications

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

Destination-State Notifications

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

- Destination Backpressure Activated
- Persistent Queue Usage
- Unhealthy Destination

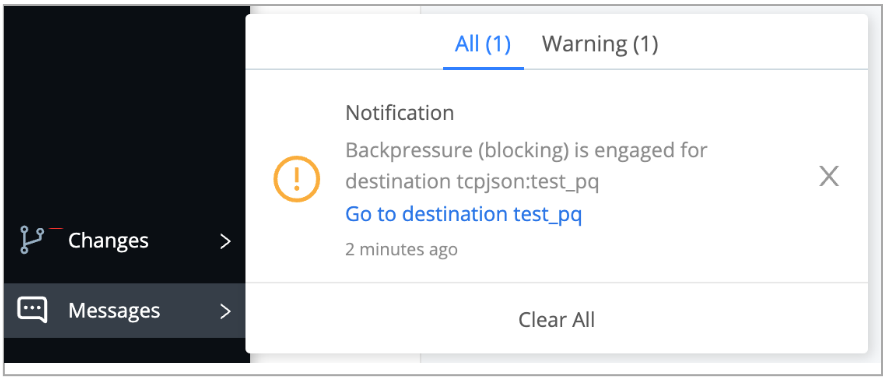
Read on for details about the above conditions, and about how to specify them in a Destination's **Condition** drop-down.

Destination Backpressure Activated

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

- The Destination's **Backpressure behavior** is set to `Block` or `Drop`, and backpressure causes outgoing events to block or drop.
- The Destination's **Backpressure behavior** is set to Persistent Queue, but its **Queue-full behavior** is set to either `Block` or `Drop new data`; and a filled queue causes outgoing events to block or drop.

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



Notification

Backpressure (blocking) is engaged for destination tcpjson:test_pq

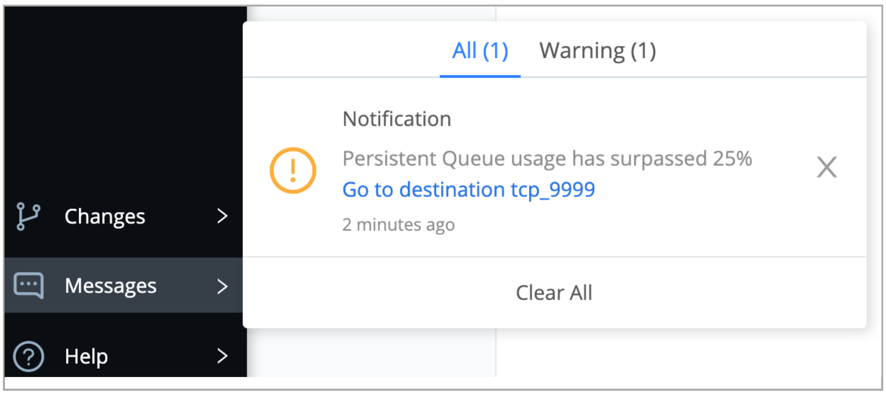
Go to destination test_pq

2 minutes ago

Backpressure notification

Persistent Queue Usage

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



Notification

Persistent Queue usage has surpassed 25%

Go to destination tcp_9999

2 minutes ago

Persistent Queue notification

Unhealthy Destination

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

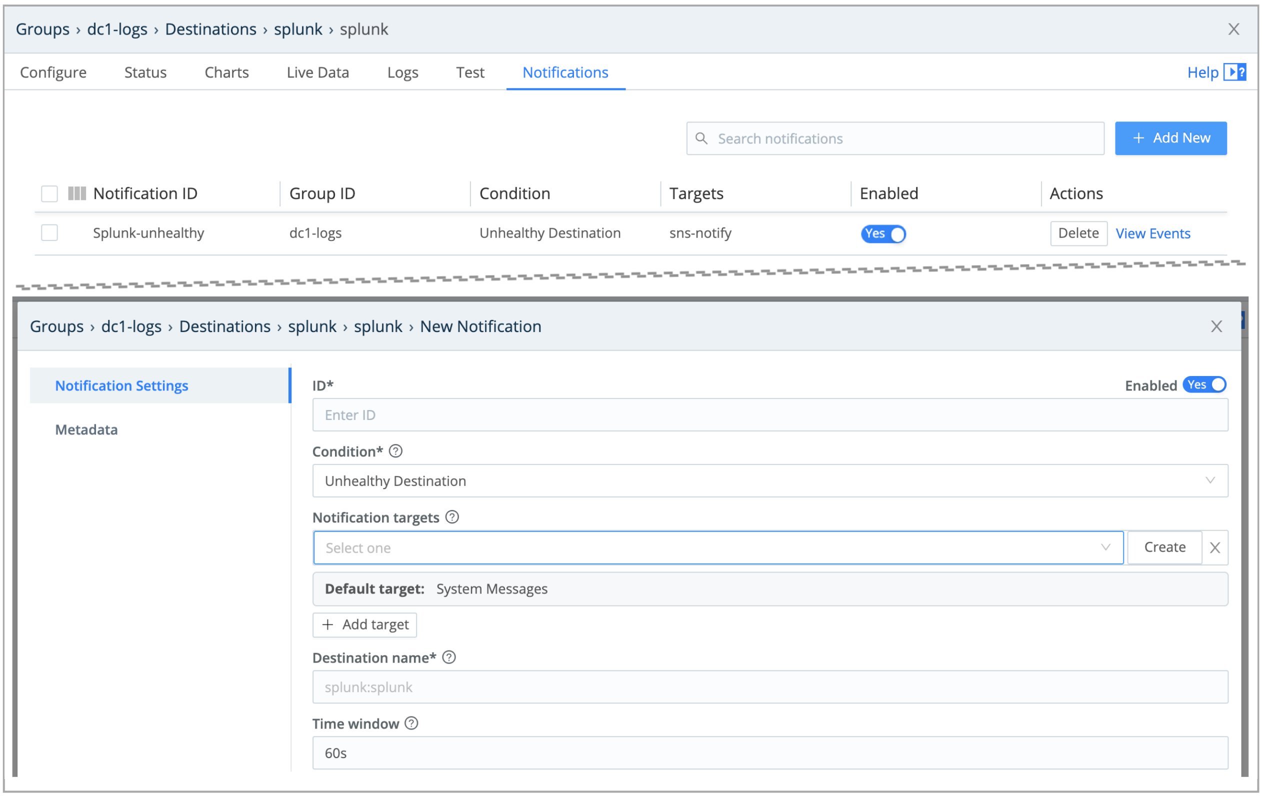
The algorithm has slight variations among Destination types, but red status generally means that ≥ 5% of health checks, aggregated over the **Time window**, reported either:

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

Configuring Destination Notifications

To start configuring a Destination-state Notification:

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



Groups › dc1-logs › Destinations › splunk › splunk

Notification ID	Group ID	Condition	Targets	Enabled	Actions
Splunk-unhealthy	dc1-logs	Unhealthy Destination	sns-notify	Yes	Delete View Events

Groups › dc1-logs › Destinations › splunk › splunk › New Notification

ID*: Enter ID
Enabled: Yes
Condition*: Unhealthy Destination
Notification targets: Select one

Default target: System Messages

Destination name*: splunk:splunk
Time window: 60s

Configuring a Destination Notification (composite screenshot)

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

Notification Settings

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

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

Notification targets: The **Default target** is always locked to `System Messages`. Click **Add target** to send this Notification to additional targets. You can add multiple targets.

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

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

Time window: Sets the threshold period before the Notification will trigger. E.g., the default `60s` will generate a Notification when the Destination has reported the trigger condition over the past 60 seconds. (For % trigger conditions, see Destination Backpressure Activated and Unhealthy Destination.) To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, etc.

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

Metadata

Click **Add field** here to add custom metadata fields to your Notifications, as key-value pairs:

Name: Enter a name for this custom field.

Value: Enter a JavaScript expression that defines this field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

> Once you've saved your Notification, you can see Notification events specific to this Destination on the Destination config modal's **Events** tab. (When you set Source-state Notifcations, a corresponding **Events** tab is available on Sources' and Collectors' config modals.) For a comprehensive view of all Notification events, see the systemwide Events Tab.

Recurrence/Expiration

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

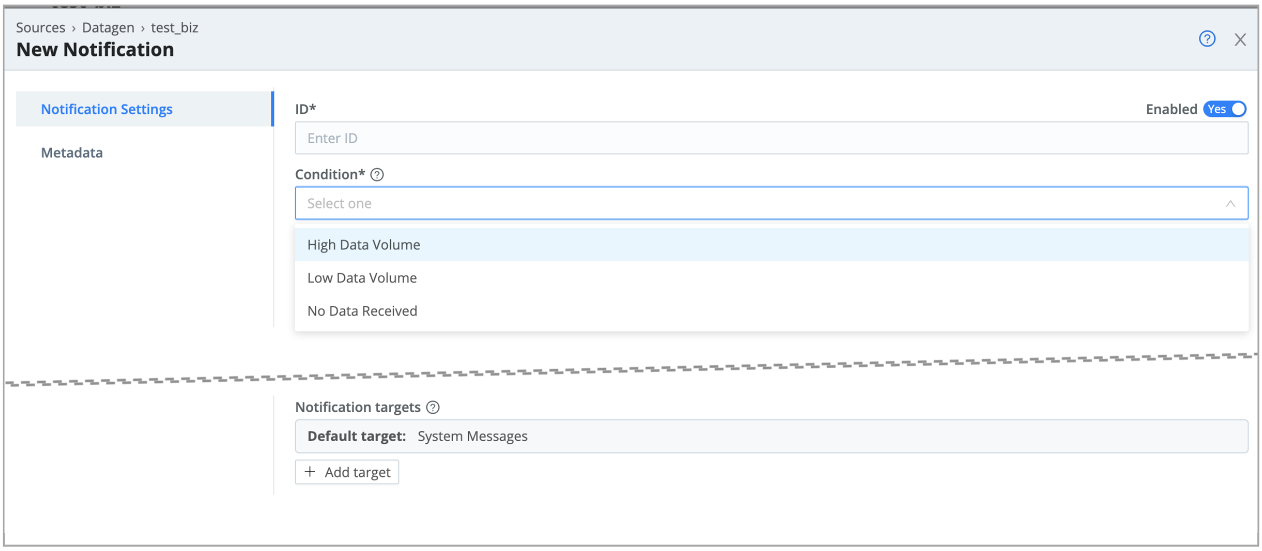
Notifications will cease when the triggering condition clears. There is no explicit "problem cleared" Notification.

Source-State Notifications

In Cribl 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

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



Sources › Datagen › test_biz

New Notification

Notification Settings

Metadata

ID*

Enabled: Yes

Enter ID

Condition*

Select one

- High Data Volume
- Low Data Volume
- No Data Received

Notification targets

Default target: System Messages

+ Add target

Configuring a High Data Volume Notification (composite screenshot)

High Data Volume

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

Notification targets: As with Destination-State Notifications, the **Default target** is always locked to `System Messages`. Click **Add target** to send this Notification to additional targets. You can add multiple targets.

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

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

Time window: As with the corresponding field on Destination-State Notifications, this field's value sets the threshold period before the Notification will trigger. The default `60s` will generate a Notification when the

Source has reported the trigger condition over the past 60 seconds. To enter alternative numeric values, append units of `s` for seconds, `m` for minutes, `h` for hours, etc.

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

Low Data Volume

Select the `Low Data Volume` Condition to trigger Notifications when incoming data over your configured **Time window** is lower than your configured **Data volume** threshold.

This selection exposes the same additional fields as High Data Volume, except that here, the **Data volume** value defines a **floor** below which the Notification will trigger.

No Data

Select the `No Data Received` Condition to trigger Notifications when the Source or Collector ingests zero data over your configured **Time window**.

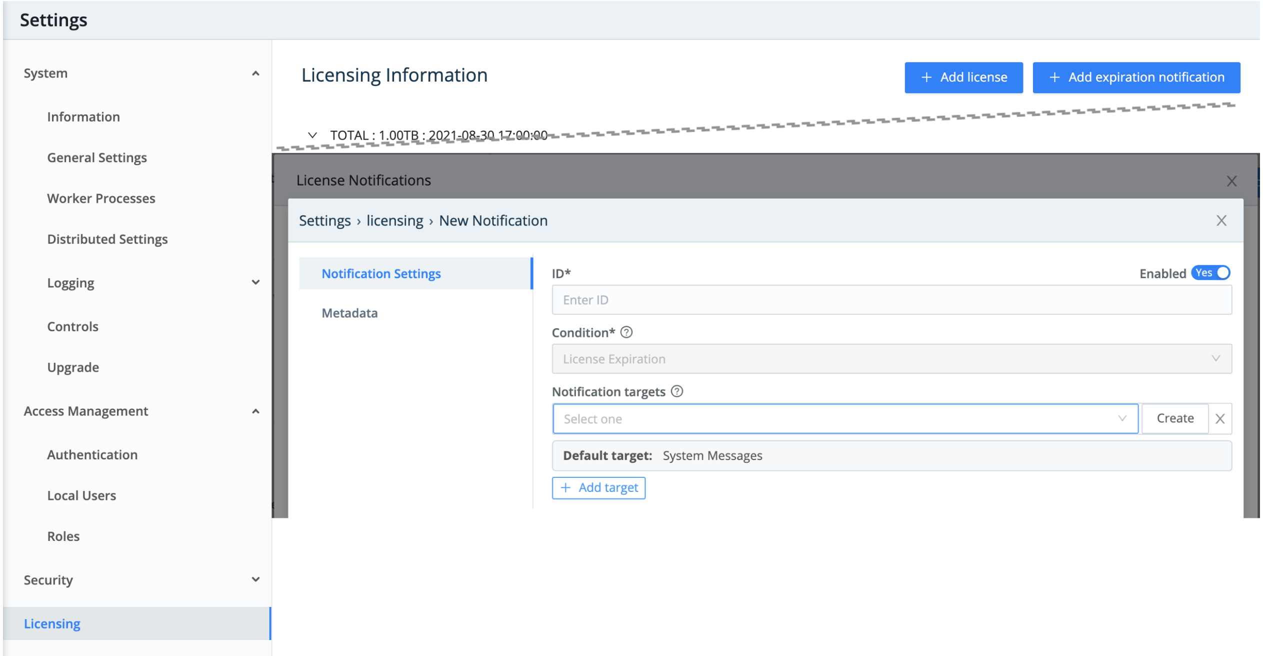
This selection exposes the same additional fields as High Data Volume, except (for obvious reasons) it omits the **Data volume** field – there is no threshold, because this is a binary condition.

License-Expiration Notifications

To prevent interruptions in data throughput, you can configure a Notification that will be triggered two weeks before your Cribl Stream paid license (licensing.#license-types) expires, and then again upon expiration. (If the two-week Notification is cleared from the 💬 **Messages** tab between those dates, but the license has not been extended, it will trigger again.)

Configuring License-Expiration Notifications

1. Select global ⚙ **Settings** (lower left) > **Licensing**.
2. Click **+ Add expiration notification** to access the **New Notification** modal shown below.



Configuring an expiration Notification (composite screenshot)

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

Notification Settings

ID: Enter a unique ID for this Notification.

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

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

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

Metadata

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

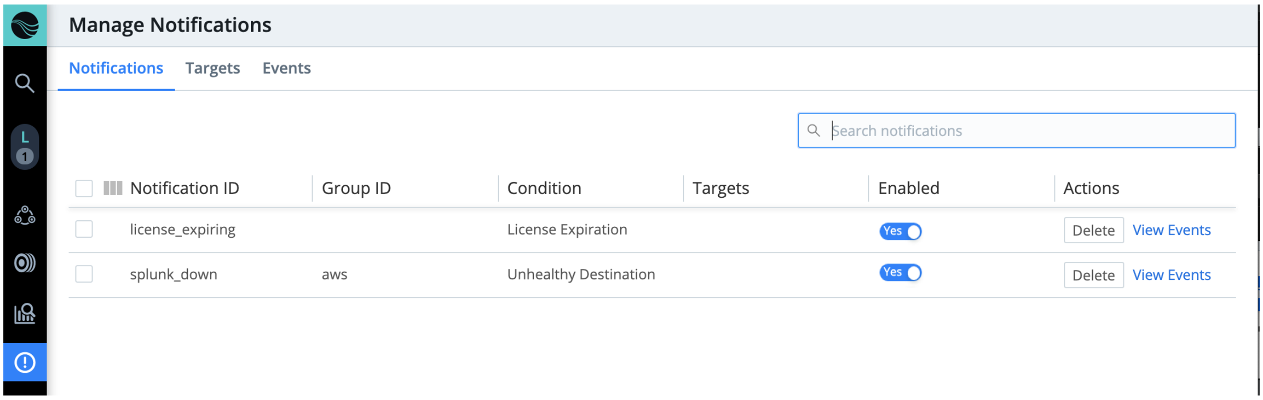
Managing Notifications

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

- In distributed deployment (Edge, Stream), click the left nav's **(!) Notifications** tab.
- In a single-instance deployment (Edge, Stream), click the top nav's **Notifications** tab.

Notifications Tab

This tab lists all your configured Source-state and Destination-unhealthy Notifications, across all integrations, along with any configured license-expiration Notifications. You can't create new Notifications here, but you can disable or delete existing Notifications; you can also click on any Notification's row to open and modify its configuration.



□	Notification ID	Group ID	Condition	Targets	Enabled	Actions
□	license_expiring		License Expiration		Yes	Delete View Events
□	splunk_down	aws	Unhealthy Destination		Yes	Delete View Events

Notifications tab

Targets Tab

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

Events Tab

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

What's Next

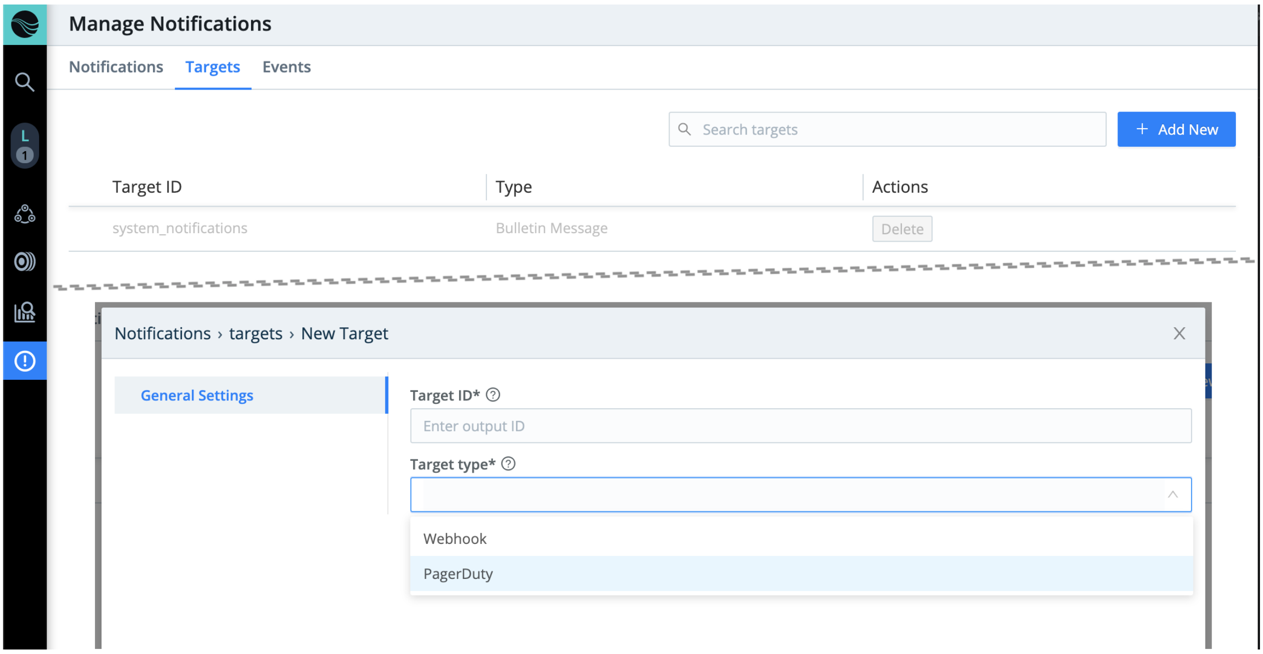
In future Cribl Stream releases, Cribl plans to expand the Notifications feature with options to configure additional triggering conditions and time resolutions.

;

4.8.1. Configuring Targets

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

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



Adding a new target (composite screenshot)

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

PagerDuty Targets

This option sends Cribl StreamNotifications to PagerDuty, a real-time incident response platform, using Cribl Stream's native integration with the PagerDuty API. Select **Target type**: `PagerDuty` to expose the following additional options on the modal's (single) **General Settings** left tab:

Routing key: Enter your 32-character Integration key on a PagerDuty service or global ruleset.

Group: Optionally, specify a PagerDuty default group to assign to Cribl 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.)

Webhook Targets

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

General Settings

The added options that appear on this first left tab are:

URL: The endpoint that should receive Cribl Stream Notification events.

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 to send to the endpoint. E.g.: `${fieldA}, ${fieldB}`. Defaults to `__httpOut` (meaning the value of the `__httpOut` field).
    - **Drop when null**: Toggle to `Yes` if you want to drop events where the above **Source expression** evaluates to `null`.
    - **Content type**: Defaults to `application/x-ndjson`. You can substitute a different content type for requests sent to the endpoint. This entry will be overridden by any content types set in this modal's **Advanced Settings** tab > **Extra HTTP Headers** section.

Processing Settings

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

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

Advanced Settings

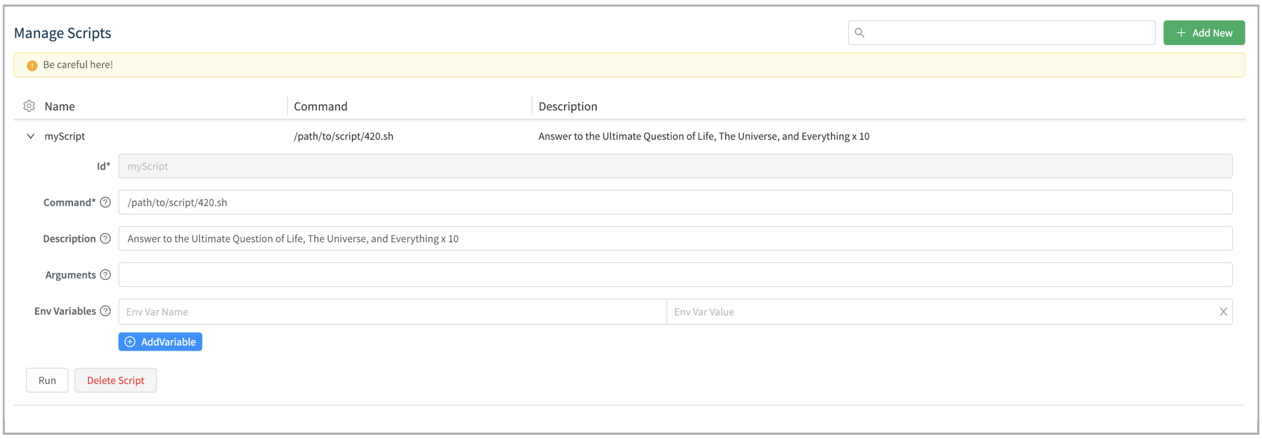
The options on this left tab are identical to those on the Webhook Destinations **Advanced Settings** tab.

;

4.9. Scripts

Admins can run scripts (e.g., shell scripts) from within Cribl Stream by configuring and executing them at global ⚙ **Settings** (lower left) > **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!



Manage Scripts

Be careful here!

Name	Command	Description
myScript	/path/to/script/420.sh	Answer to the Ultimate Question of Life, The Universe, and Everything x 10

Id*	myScript
Command*	/path/to/script/420.sh
Description	Answer to the Ultimate Question of Life, The Universe, and Everything x 10
Arguments	
Env Variables	Env Var Name / Env Var Value

Settings > Manage Scripts page

The Manage Scripts page provides the following tields:

- **ID**: Unique ID for this script.
- **Command**: Command to execute for this script.
- **Description**: Brief description about this script. Optional.
- **Arguments**: Arguments to pass when executing this script.
- **Env variables**: Extra environment variables to set when executing script.

> **Scripts in Distributed Deployments**
>
> - Scripts can be deployed from Leader Node, but can be **run** only locally from each Worker Node.
> - If the Script command is referencing a file (e.g., `420.sh`), that file must exist on the Cribl Stream instance. In other words, the Script management interface cannot be used to upload or manage

> script files.

;

4.10. Upgrading

This page outlines how to upgrade a Cribl Stream single-instance or distributed deployment along one of the following supported upgrade paths:

- v2.x ==> v2.x || v3.x
- v1.7.x/v2.0.x ==> v2.x.x || v3.x
- v1.6.x or below ==> v1.7.x ==> v2.x.x || v3.x

> Cribl Stream does **not** support direct upgrades from a Beta to a GA version. To get the GA version running, you must perform a new install. See notes on Upgrading from LogStream 2.2 or Prior Versions below.

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, the general order of upgrade is: First upgrade the Leader Node, then upgrade the Worker Nodes, then commit and deploy the changes on the Leader.

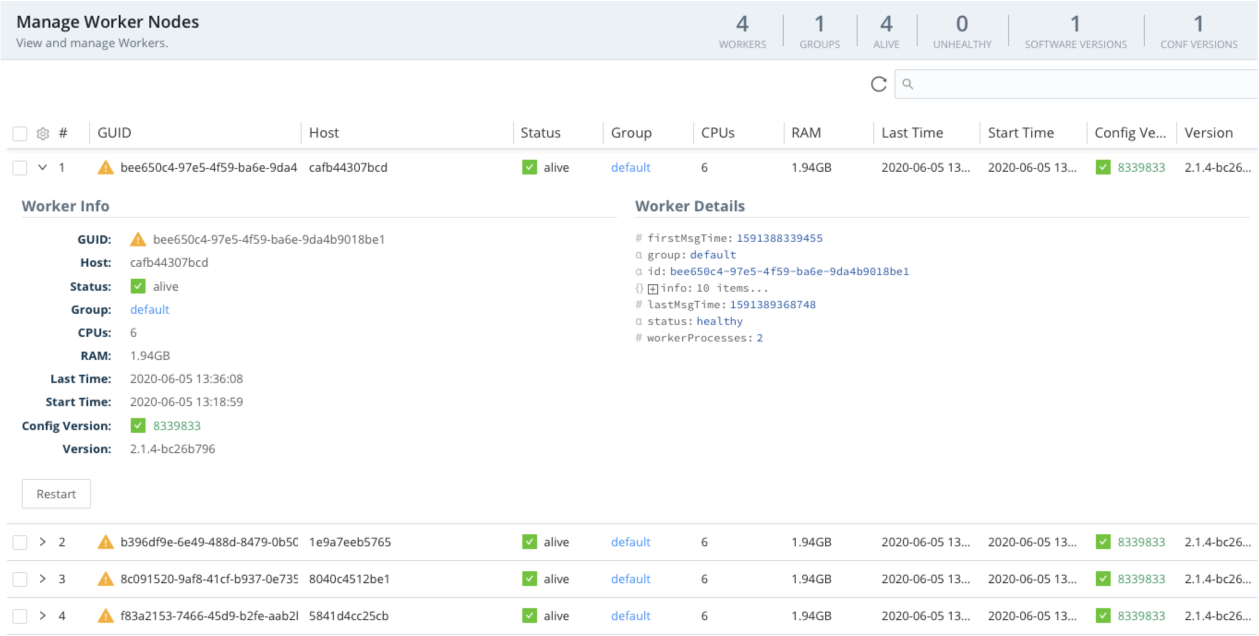
> This sequence is slightly different when upgrading via the UI from a version below v.3.2.

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

Upgrade the Leader Node

1. Commit and deploy your desired last version. (This will be your most recent checkpoint.)
   - Optionally, `git push` to your configured remote repo.
2. Stop Cribl 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. Restart Cribl Stream and log back in.
5. Wait for all the Worker Nodes to report to the Leader, and ensure that they are correctly reporting the last committed configuration version.

> Workers' UI will not be available until the Worker version has been upgraded to match the version on the Leader. Errors like those below will appear until the Worker nodes are upgraded.



Manage Worker Nodes
View and manage Workers.

4 WORKERS | 1 GROUPS | 4 ALIVE | 0 UNHEALTHY | 1 SOFTWARE VERSIONS | 1 CONF VERSIONS

#	GUID	Host	Status	Group	CPUs	RAM	Last Time	Start Time	Config Ve...	Version
1	bee650c4-97e5-4f59-ba6e-9da4	cafb44307bcd	✓ alive	default	6	1.94GB	2020-06-05 13...	2020-06-05 13...	✓ 8339833	2.1.4-bc26...
2	b396df9e-6e49-488d-8479-0b5C	1e9a7eeb5765	✓ alive	default	6	1.94GB	2020-06-05 13...	2020-06-05 13...	✓ 8339833	2.1.4-bc26...
3	8c091520-9af8-41cf-b937-0e735	8040c4512be1	✓ alive	default	6	1.94GB	2020-06-05 13...	2020-06-05 13...	✓ 8339833	2.1.4-bc26...
4	f83a2153-7466-45d9-b2fe-aab2l	5841d4cc25cb	✓ alive	default	6	1.94GB	2020-06-05 13...	2020-06-05 13...	✓ 8339833	2.1.4-bc26...

Worker Info

GUID: bee650c4-97e5-4f59-ba6e-9da4b9018be1
Host: cafb44307bcd
Status: ✓ alive
Group: default
CPUs: 6
RAM: 1.94GB
Last Time: 2020-06-05 13:36:08
Start Time: 2020-06-05 13:18:59
Config Version: ✓ 8339833
Version: 2.1.4-bc26b796

Worker Details

```
# firstMsgTime: 1591388339455
a group: default
a id: bee650c4-97e5-4f59-ba6e-9da4b9018be1
{} ⊞info: 10 items...
# lastMsgTime: 1591389368748
a status: healthy
# workerProcesses: 2
```

Worker Node version mismatch

Upgrade the Worker Nodes

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

1. Stop Cribl Stream on each Worker Node.

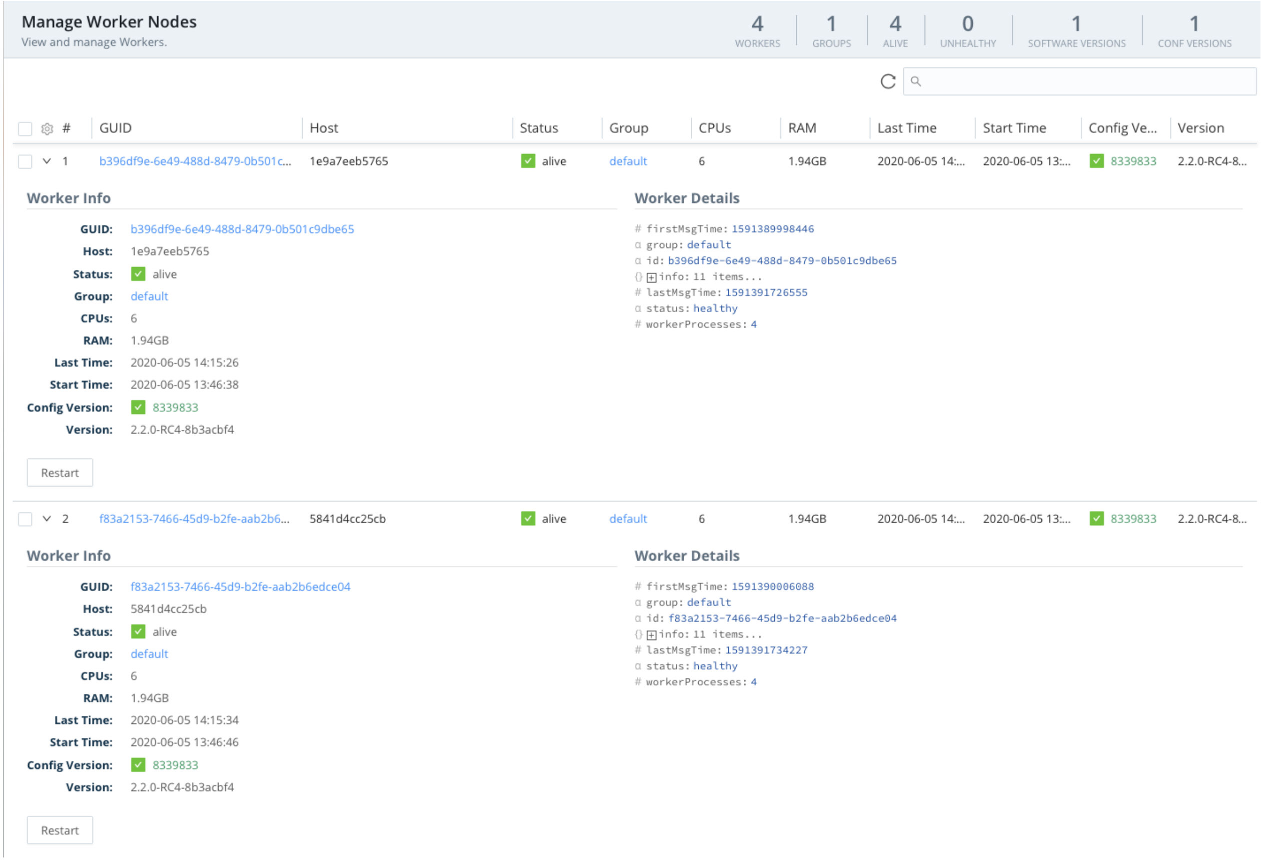
2. Uncompress the new version on top of the old one.

3. Restart Cribl Stream.

Commit and Deploy 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 configuration **only after all** Workers have upgraded.



Manage Worker Nodes
View and manage Workers.

4	1	4	0	1	1
WORKERS	GROUPS	ALIVE	UNHEALTHY	SOFTWARE VERSIONS	CONF VERSIONS

#	GUID	Host	Status	Group	CPUs	RAM	Last Time	Start Time	Config Ve...	Version
1	b396df9e-6e49-488d-8479-0b501c...	1e9a7eeb5765	✓ alive	default	6	1.94GB	2020-06-05 14:...	2020-06-05 13:...	✓ 8339833	2.2.0-RC4-8...

Worker Info

GUID: b396df9e-6e49-488d-8479-0b501c9dbe65
Host: 1e9a7eeb5765
Status: ✓ alive
Group: default
CPUs: 6
RAM: 1.94GB
Last Time: 2020-06-05 14:15:26
Start Time: 2020-06-05 13:46:38
Config Version: ✓ 8339833
Version: 2.2.0-RC4-8b3acbf4

Worker Details

```
# firstMsgTime: 1591389998446
a group: default
a id: b396df9e-6e49-488d-8479-0b501c9dbe65
{} ⊞info: 11 items...
# lastMsgTime: 1591391726555
a status: healthy
# workerProcesses: 4
```

#	GUID	Host	Status	Group	CPUs	RAM	Last Time	Start Time	Config Ve...	Version
2	f83a2153-7466-45d9-b2fe-aab2b6...	5841d4cc25cb	✓ alive	default	6	1.94GB	2020-06-05 14:...	2020-06-05 13:...	✓ 8339833	2.2.0-RC4-8...

Worker Info

GUID: f83a2153-7466-45d9-b2fe-aab2b6edce04
Host: 5841d4cc25cb
Status: ✓ alive
Group: default
CPUs: 6
RAM: 1.94GB
Last Time: 2020-06-05 14:15:34
Start Time: 2020-06-05 13:46:46
Config Version: ✓ 8339833
Version: 2.2.0-RC4-8b3acbf4

Worker Details

```
# firstMsgTime: 1591390006088
a group: default
a id: f83a2153-7466-45d9-b2fe-aab2b6edce04
{} ⊞info: 11 items...
# lastMsgTime: 1591391734227
a status: healthy
# workerProcesses: 4
```

Post-2.1.4 upgrade to 2.2

Upgrade and Rollback via the UI

LogStream v.2.4.4 and higher provide streamlined options to upgrade the Leader Node (or single instance), as well as Worker Nodes, directly through the UI. In LogStream 3.0 or higher, go to global ⚙ **Settings** (lower left) > **System > Upgrade**. These streamlined controls perform the whole above sequence of stopping the LogStream server, updating the installed package, and restarting LogStream.

LogStream 3.0 (or higher) also enables you to manage automatic backup and rollback in case an upgrade fails.

> These options will work only if all LogStream instances (including Worker Processes) start at v.2.4.4 or higher. For other version-specific limitations, please see Upgrading to v.3.2 and Beyond below and Known Issues.
>
> Be aware that the **Checking for upgrade** status message, and its accompanying spinner, can take up to several minutes to resolve. Also, after you initiate an upgrade, it can take up to several minutes before the **View** button (described below) is displayed.

The following controls are available here:

Package Source

Beside **Package source**, the default **CDN** button downloads a package directly from Cribl's content delivery network.

If you select the alternative **Path** button, next click **+ Add Path** (below **Custom path**) to define as many paths as you need. In the resulting table, each row provides these additional fields:

- **Platform-specific package location**: Enter either a URL (HTTP) or a local path to the upgrade package.
- **Package hash location**: Enter either a URL (HTTP), or a local 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.)
- **X** (close box): Click to delete a row – immediately. (There is no confirmation prompt.)
- **Save**/**Cancel** buttons: Click **Save** to store the specified locations. Clicking **Cancel** restores the **CDN** package-source selection.

Different rows in this table can identify packages for different platforms/architectures.

Automatic Upgrades

The **Disable automatic upgrades** slider has a different default state per deployment type:

In on-prem/customer-managed deployments, **Disable automatic upgrades** defaults to `Yes`, to prevent Cribl Stream from automatically upgrading out-of-date Worker Nodes. Before you toggle this to `No` (i.e., enable automatic upgrades) in a self-managed deployment, see Upgrading to v.3.2 and Beyond below.

In Cribl.Cloud deployments, **Disable automatic upgrades** defaults to `No`. 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 `Yes`, you will need to explicitly upgrade each Worker. Cribl-managed and hybrid Workers will auto-upgrade as soon as they see a new Leader version. If you toggle this to `Yes`, you will need to explicitly upgrade each Worker.

When enabled, the automatic 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 vs. Upgrade Leader

In a Single-instance deployment, the **Upgrade** button is the only other control provided. In a distributed deployment, the **Upgrade** button is displayed on an **Upgrade Leader** tab, and clicking it upgrades the Leader Node. (As with manual upgrades, always upgrade the Leader before upgrading the Workers.)

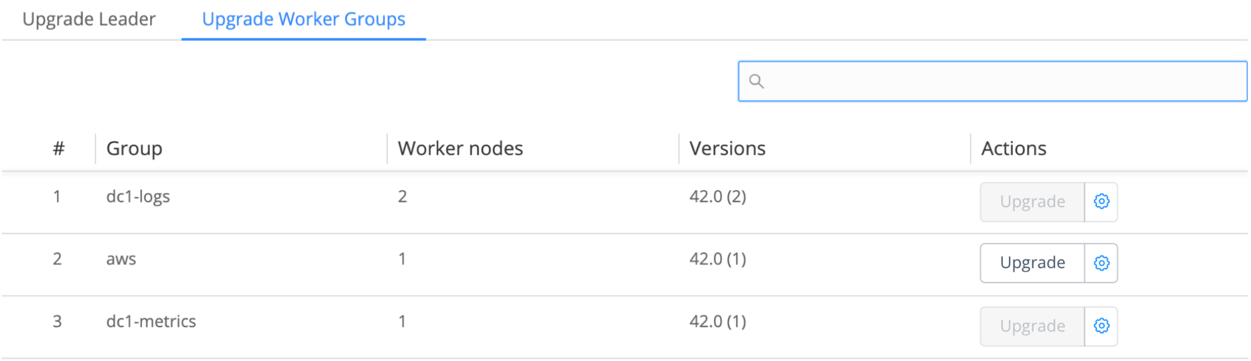
Upgrading to v.3.2 and Beyond

When using the UI to upgrade to v.3.2 or higher, commit and deploy your upgrade on the Leader before using the following options to upgrade Workers to a corresponding version. This is especially important if you enable automatic upgrades.

> ### 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 LogStream 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+.

Upgrade Worker Groups

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



#	Group	Worker nodes	Versions	Actions
1	dc1-logs	2	42.0 (2)	Upgrade
2	aws	1	42.0 (1)	Upgrade
3	dc1-metrics	1	42.0 (1)	Upgrade

Upgrade Worker Groups tab

> Upgrading Workers from the Leader requires a Cribl Stream Standard or Enterprise license.

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

Basic Upgrade Configuration

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

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.

Advanced Upgrade Configuration

Click **Advanced configuration** 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: Toggle this slider on to upgrade Workers one at a time. Enabling the slider also enables the dialog's two remaining controls:

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

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

After you click the **Upgrade** confirmation button, the **Upgrade Worker Groups** tab 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

By default, Cribl Stream will automatically roll back to a stored backup package if an upgrade (initiated through the UI) fails. You can adjust this behavior at global ⚙ **Settings** (lower left) > **System > General Settings > Upgrade & Share Settings**, using the following controls.

> LogStream can perform rollbacks only on Worker Nodes/instances that started on at least LogStream 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`.

Upgrading from LogStream 2.2 or Prior Versions

As of version 2.3, the LogStream Free license is permanent, but it enforces certain restrictions that especially affect distributed deployments:

- Even if you have more than one Worker Group defined, only one Worker Group will be visible and usable.
  - This will be the first Group listed in `$CRIBL_HOME/local/cribl/groups.yml` – typically, the `default` Group. You can edit `groups.yml` to move the desired Group to the top.
- Your cluster will be limited to 10 Worker Processes across all Worker Nodes.
  - Cribl Stream will balance (or rebalance) these Processes as evenly as possible across the Worker Nodes.
- Authentication will fall back to local authorization. You will not be able to authenticate via Splunk, LDAP, or SSO/OpenID.
- **Git Push** to remote repos will not be supported through the product.

> If you are upgrading LogStream Free from version 2.2.x or lower, these changes might require you to adjust your existing configuration and/or workflows.
>
> See Licensing for details on all current license options.

As of LogStream 2.3, licenses no longer need to be deployed directly to Worker Groups. The Leader will push license information down to Worker Groups as part of the heartbeat.

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.

;

4.11. 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.

- Remove the directory where Cribl Stream is installed.

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).

;

5. Working With Data

5.1. 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`.

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):

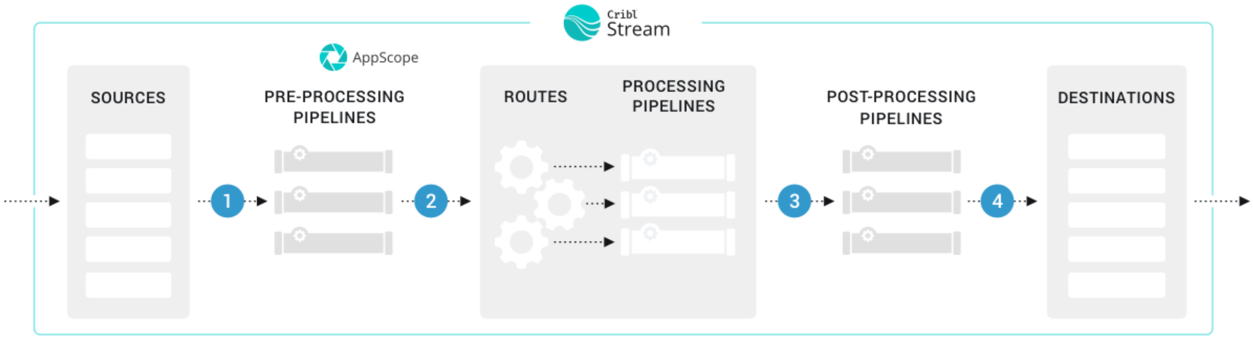
1. Click **Start a Capture**.

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:

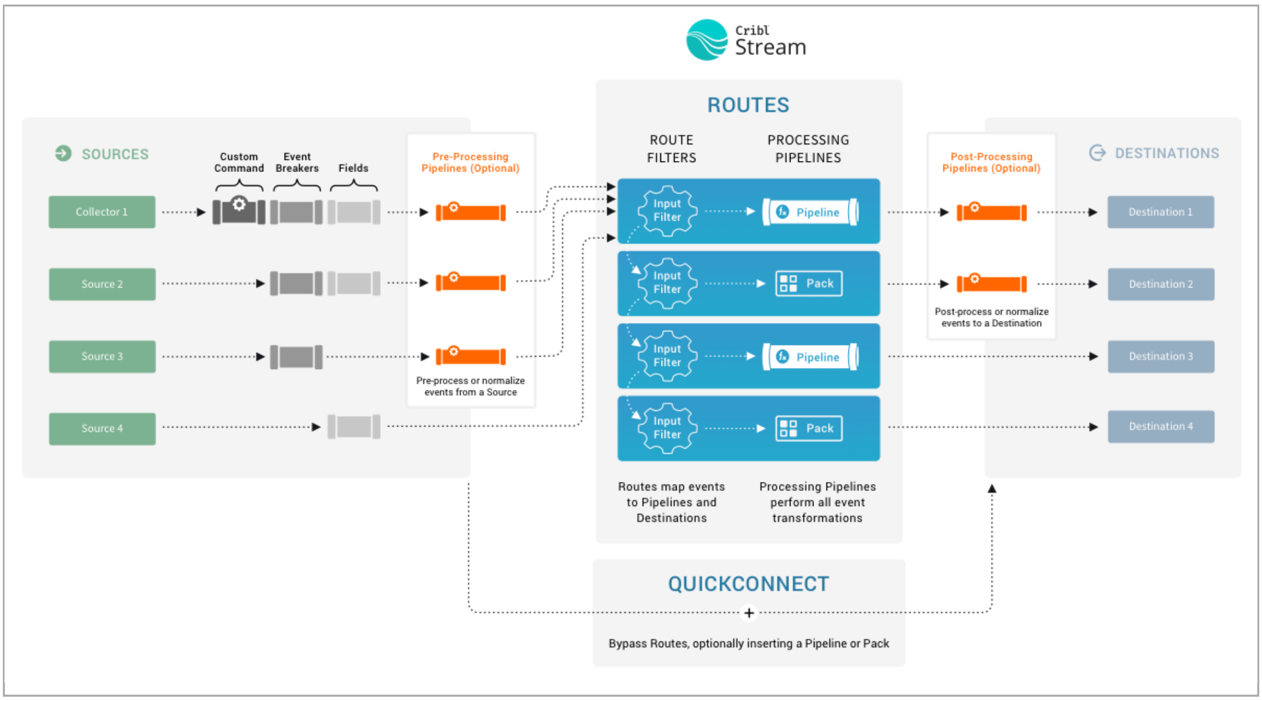
- **1. Before the pre-processing Pipeline** – Capture events right after they're delivered by the respective Input.
- **2. Before the Routes** – Capture events right after the pre-processing Pipeline. before they go down any Routes. (QuickConnect bypasses Routes, and can bypass processing Pipelines.)
- **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.



;

5.2. 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
SOURCES
Custom Command
Event Breakers
Fields
Collector 1
Source 2
Source 3
Source 4
Pre-Processing Pipelines (Optional)
Pre-process or normalize events from a Source
ROUTES
ROUTE FILTERS
PROCESSING PIPELINES
Input Filter
Pipeline
Input Filter
Pack
Input Filter
Pipeline
Input Filter
Pack
Routes map events to Pipelines and Destinations
Processing Pipelines perform all event transformations
Post-Processing Pipelines (Optional)
Post-process or normalize events to a Destination
DESTINATIONS
Destination 1
Destination 2
Destination 3
Destination 4
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. Pre-processing Pipeline: Optionally, you can use a single Pipeline to condition (normalize) data from this input before the data proceeds further.

8. Routes map incoming events to Processing Pipelines and Destinations. A Route can accept data from multiple Sources, but each Route can be associated with only one Pipeline and one Destination. (QuickConnect bypasses Routes.)

9. 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.)

10. Post-processing Pipeline: Optionally, you can append a Pipeline to condition (normalize) data on its way into its Destination.

11. 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.

;

5.3. 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 **+ Route**.

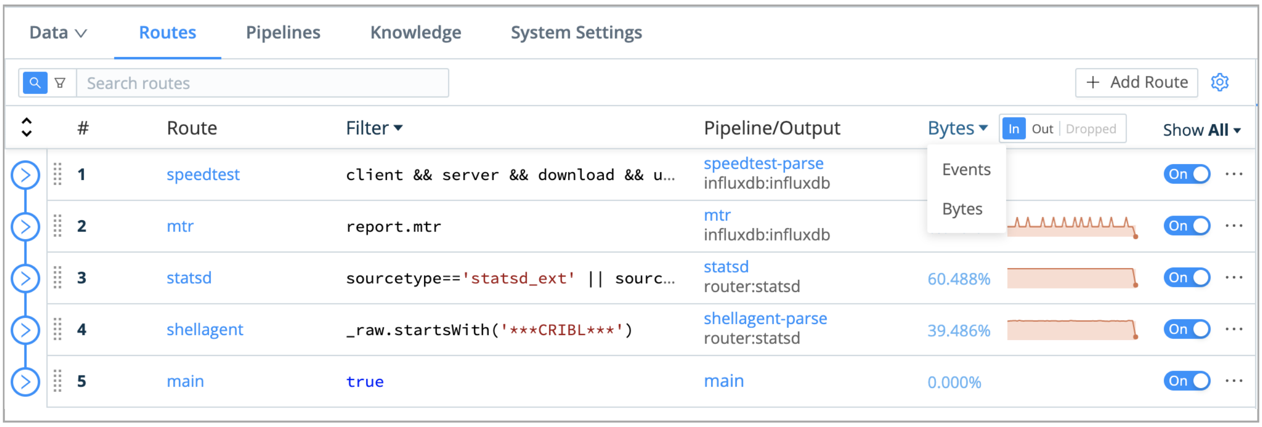
How Do Routes Work

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** Pipeline.

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.



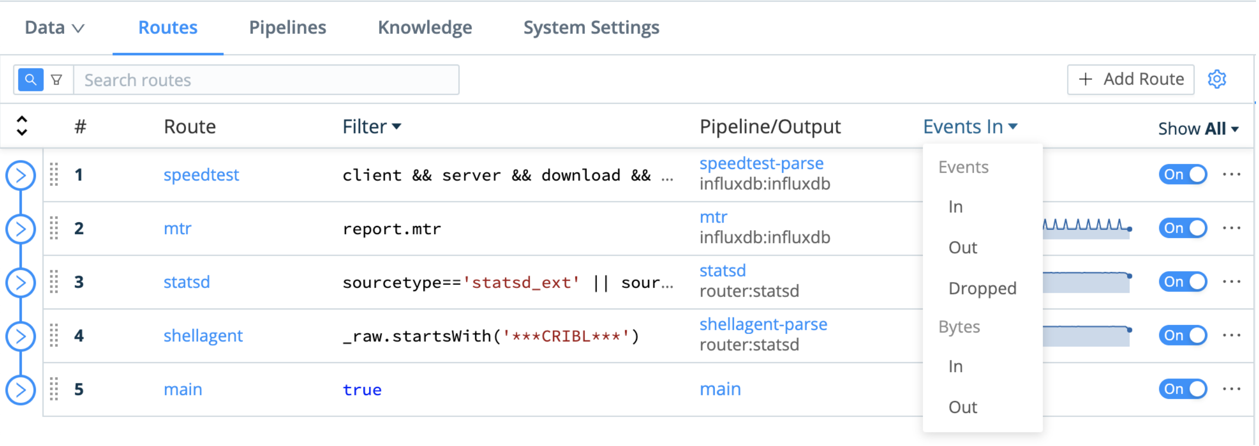
#	Route	Filter	Pipeline/Output	Bytes
1	speedtest	`client && server && download && u…`	speedtest-parse influxdb:influxdb	
2	mtr	`report.mtr`	mtr influxdb:influxdb	
3	statsd	`sourcetype=='statsd_ext' || sourc…`	statsd router:statsd	60.488%
4	shellagent	`_raw.startsWith('***CRIBL***')`	shellagent-parse router:statsd	39.486%
5	main	`true`	main	0.000%

Routes and bytes

In the example above, incoming events will be evaluated first against the Route named **speedtest**, then against **mtr**, then against **statsd**, and so on. At the end, the **main** Route serves as a catch-all for any event that does not match any of the other Routes.

Above, note the selectors to toggle between displaying Events versus Bytes, and to display In versus Out.

When you condense the Routes page to a narrower viewport, Cribl Stream consolidates the **In**/**Out**/**Dropped** selectors onto an expanded **Bytes**/**Events** drop-down menu, as shown below.



#	Route	Filter	Pipeline/Output	Events In
1	speedtest	`client && server && download && …`	speedtest-parse influxdb:influxdb	
2	mtr	`report.mtr`	mtr influxdb:influxdb	
3	statsd	`sourcetype=='statsd_ext' || sour…`	statsd router:statsd	
4	shellagent	`_raw.startsWith('***CRIBL***')`	shellagent-parse router:statsd	
5	main	`true`	main	

Routes and events (combined menu)

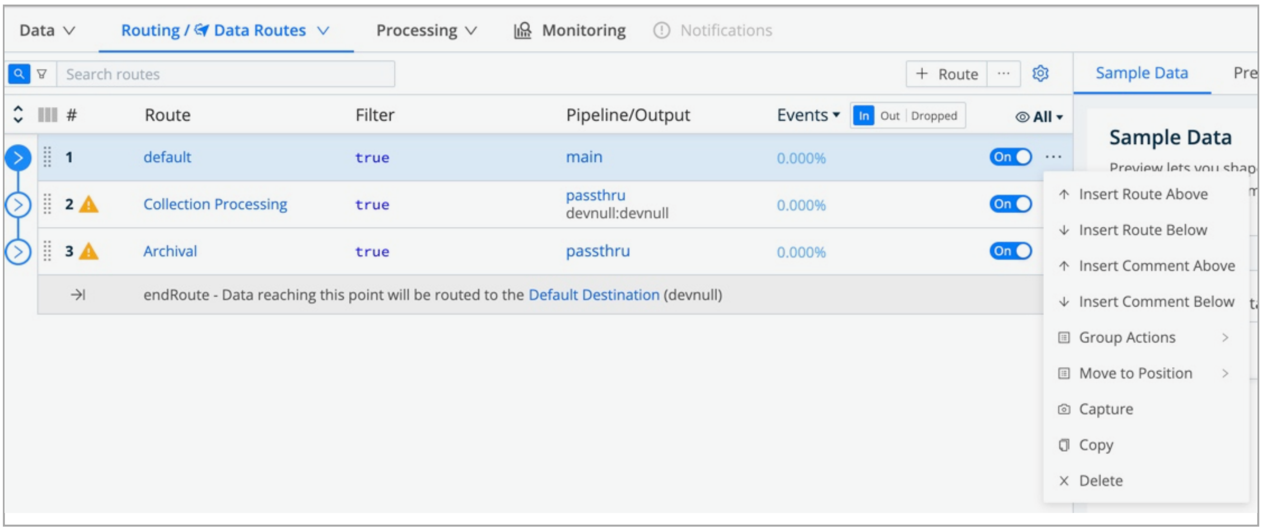
Managing the Routes Page

To apply a Route before another, simply drag it vertically. Use the sliders to turn Routes **On**/**Off** inline, as necessary, to facilitate development and debugging.

You can press the `]` (right-bracket) shortcut key to toggle between the Preview pane and the expanded Routes display shown above. (This works when no field has focus.)

Click a Route's Options (**...**) menu to display multiple options:

- Insert, group, move, copy, or delete Routes.
- Insert comments above or below Routes.
- Capture sample data through a selected Route.

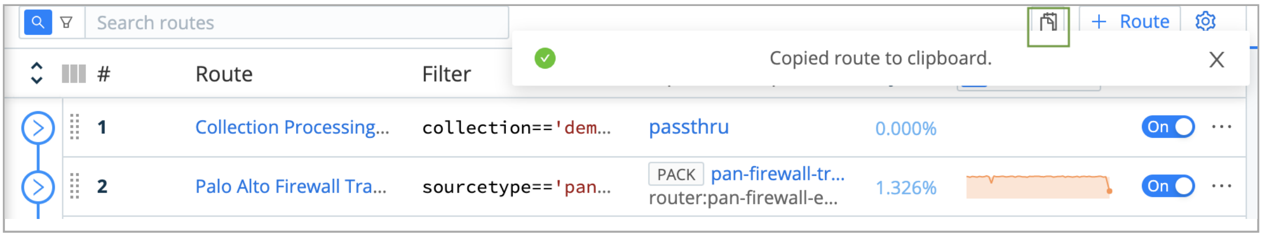


#	Route	Filter	Pipeline/Output	Events	
1	default	true	main	0.000%	On
2	Collection Processing	true	passthru devnull:devnull	0.000%	On
3	Archival	true	passthru	0.000%	On

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

Route > Options menu

Copying a Route displays the confirmation message and the (highlighted) Paste button shown below.

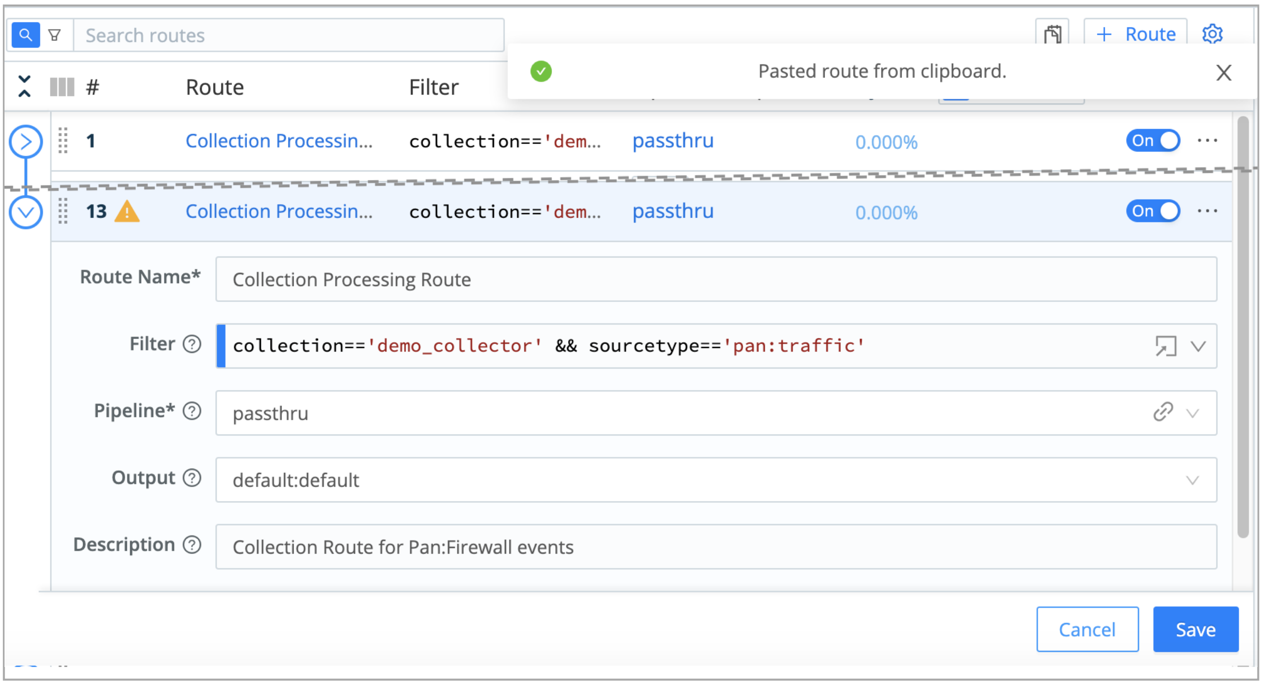


Copied route to clipboard.

#	Route	Filter			
1	Collection Processing...	collection=='dem...	passthru	0.000%	On
2	Palo Alto Firewall Tra...	sourcetype=='pan...	PACK pan-firewall-tr... router:pan-firewall-e...	1.326%	On

Paste button for copied Route

Pasting creates an exact duplicate of the Route, with a warning indicator to change its duplicate name.



Pasted route from clipboard.

#	Route	Filter	Pipeline	Bytes	
1	Collection Processin...	collection=='dem...	passthru	0.000%	On
13	Collection Processin...	collection=='dem...	passthru	0.000%	On

Route Name*: Collection Processing Route

Filter: `collection=='demo_collector' && sourcetype=='pan:traffic'`

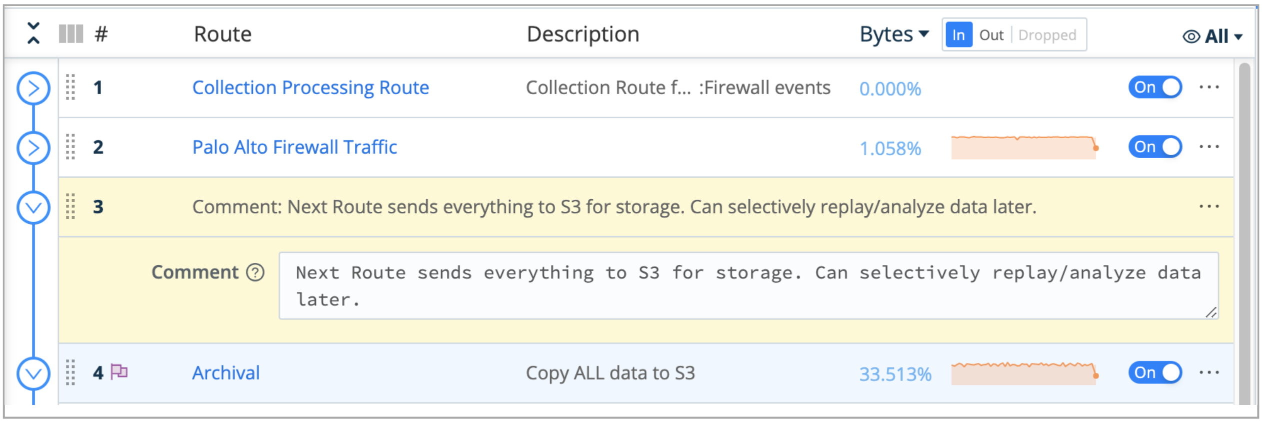
Pipeline*: passthru

Output: default:default

Description: Collection Route for Pan:Firewall events

Pasted duplicate Route

Comments work like a Pipeline's Comment Functions: You can use them to describe Routes' purposes and/or interactions.



#	Route	Description	Bytes	
1	Collection Processing Route	Collection Route f... :Firewall events	0.000%	On
2	Palo Alto Firewall Traffic		1.058%	On
3	Comment: Next Route sends everything to S3 for storage. Can selectively replay/analyze data later.			
4	Archival	Copy ALL data to S3	33.513%	On

Comment: Next Route sends everything to S3 for storage. Can selectively replay/analyze data later.

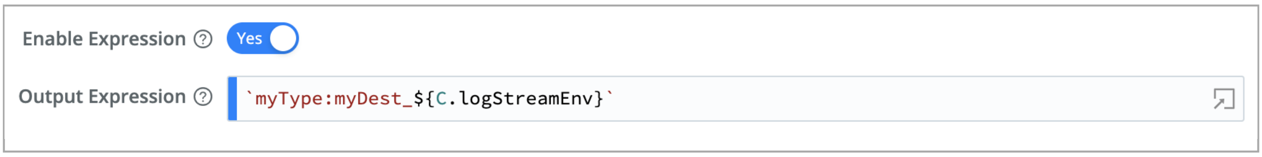
Comments make the Routing table self-documenting

Output Destination

You can configure each Route with an **Output** that defines a Destination to send events to, after they're processed by the Pipeline.

If you toggle the **Enable expression** slider to `Yes`, the **Output** field changes to an **Output expression** field. Here, you can enter a JavaScript expression that Cribl Stream will evaluate as the name of the Destination.

Sample expression format: `myType:myDest_${C.logStreamEnv}`. (This evaluation happens at Route construction time, not per event.)



Enable Expression ⓘ Yes

Output Expression ⓘ `` `myType:myDest_${C.logStreamEnv}` ``

Output expression enabled

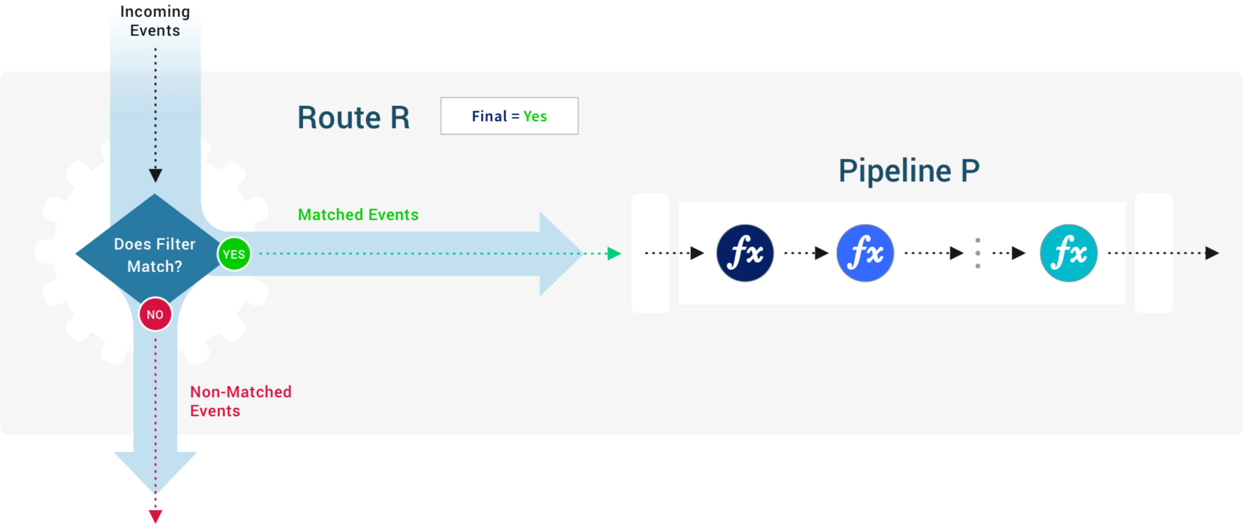
> Expressions that match no Destination name will silently fail.
>
> On Routes within Packs, neither **Output** control has any effect, because Packs cannot specify Destinations.

The `Final` Toggle

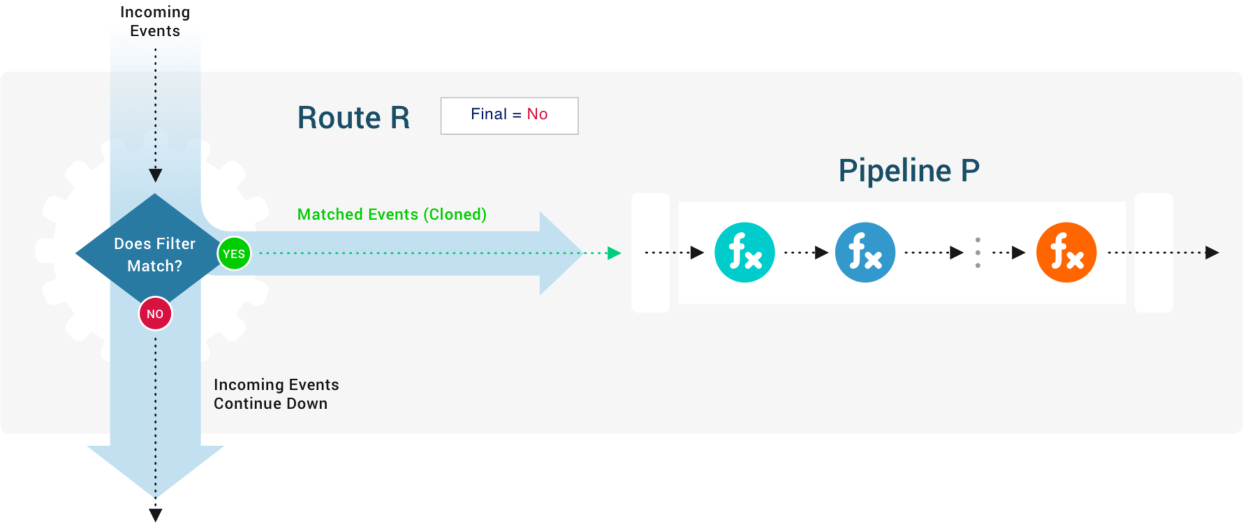
When an event that enters the system and matches a Route-Pipeline pair, it will usually be either:

- Dropped by a function, or
- Transformed (optionally) and exit the system.

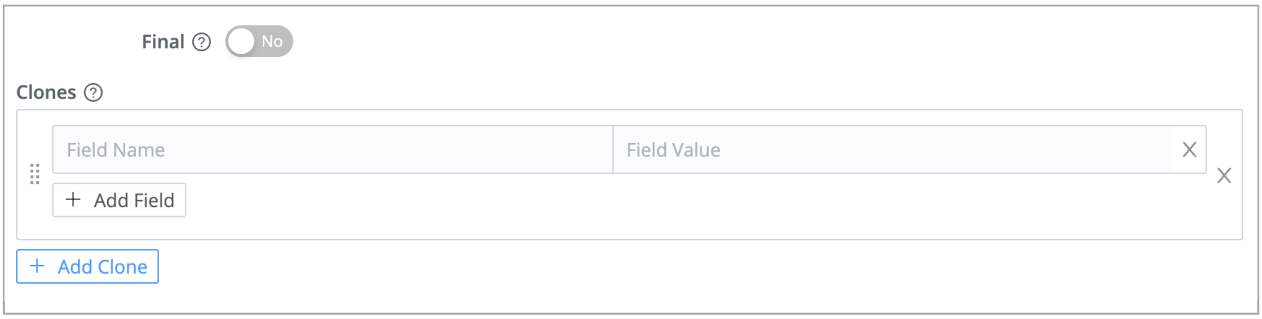
This behavior is ensured by the `Final` toggle in Route settings. It defaults to `Yes`, meaning that matched events will be **consumed** by that Route, and will not be evaluated against any other Routes that sit below it.



If the `Final` toggle is set to `No`, clone(s) of the matching events will be processed by the configured Pipeline, and the original events will be allowed to continue their trip to be evaluated and/or processed by other Route-Pipeline pairs.



When you set the `Final` toggle to `No`, the **Add Clone** button appears. Click it to open the table shown below. Here, you can add a field – name and value – to the cloned events sent to this Route's Pipeline. Click **Add Field** to define more added fields/values.



Non-final Route: add clone fields

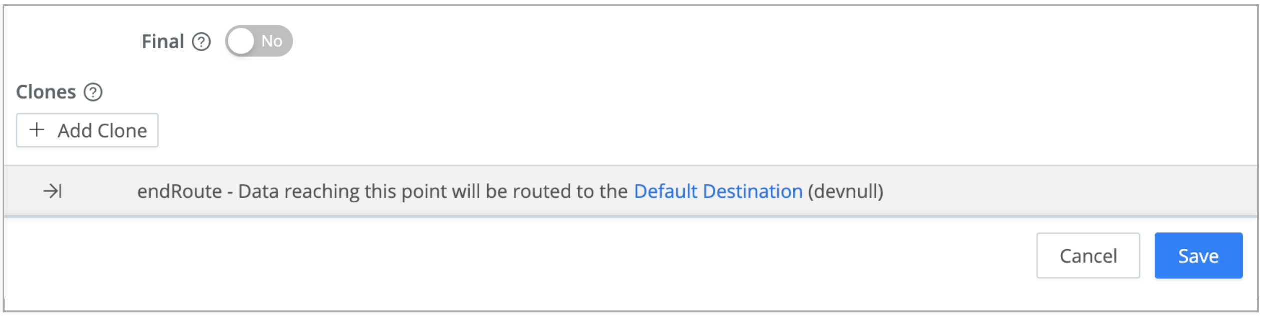
Final Flag and Cloning Considerations

Depending on your cloning needs, you might want to follow a **most-specific first** or a **most-general first** processing strategy. The general goal is to minimize the number of filters/Routes an event gets evaluated against. For example:

- If cloning is not needed at all (i.e., all `Final` toggles stay at default), then it makes sense to start with the broadest expression at the top, so as to consume as many events as early as possible.
- If cloning is needed on a narrow set of events, then it might make sense to do that upfront, and follow it with a Route that consumes those clones immediately after.

The `endRoute` Bumper

The `endRoute` bumper appears at the bottom of the Routing table. This is a reminder that, if **no** Route in the table has the `Final` flag enabled, events will continue to Cribl Stream's configured Default Destination.



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

endRoute warning and link

This is a backstop to ensure data flow. However, if that configured default is also configured as the **Output** of a Route higher in the table, duplicate events will reach that Destination.

You can correct this either by setting a Route to `Final`, or by changing the Default Destination. The bumper provides a link to the Default Destination's config, and identifies the currently configured default in parentheses.

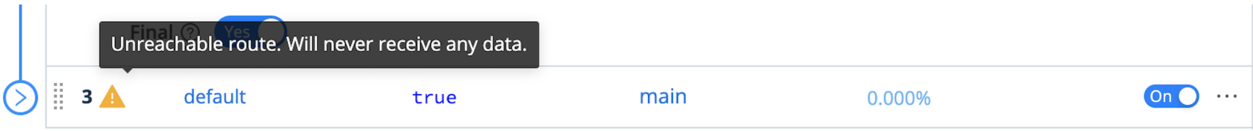
Route Groups

A Route group is a collection of consecutive Routes that can be moved up and down the Route stack together. Groups help with managing long lists of Routes. They are a UI visualization only: While Routes are in a group, those Routes maintain their global position order.

> Route groups work much like Function groups, offering similar UI controls and drag-and-drop options.

Unreachable Routes

Routes display an "unreachable" warning indicator (orange triangle) when data can't reach them.


Unreachable route. Will never receive any data.

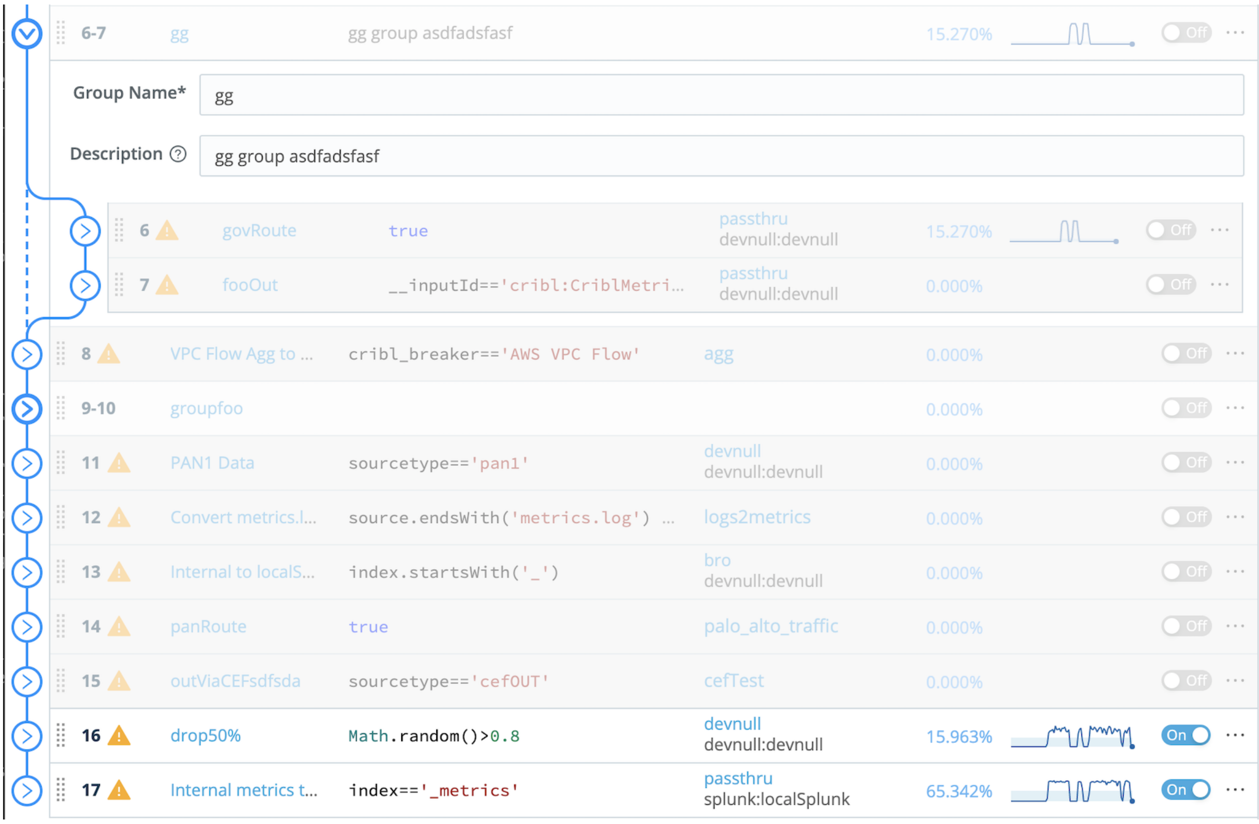
3 ⚠	default	`true`	main	0.000%	On

Unreachable Route warning, on hover

This condition will occur when, with your current configuration, any Route higher in the stack matches **all** three of these conditions:

- Previous Route is enabled (slider is set to `On`).
- Previous Route is final (**Final** slider is set to `Yes`).
- Previous Route's **Filter** expression evaluates to true, (e.g., `true`, `1 === 1`, etc.).

Note that the third condition above can be triggered intermittently by a randomizing method like `Math.random()`. This might be included in a previous Route's own Filter expression, or in a Pipeline Function (such as one configured for random data sampling).



6-7	gg	gg group asdfadsfasf		15.270%	Off

Group Name* gg

Description gg group asdfadsfasf

6 ⚠	govRoute	`true`	passthru devnull:devnull	15.270%	Off
7 ⚠	fooOut	`__inputId=='cribl:CriblMetri...`	passthru devnull:devnull	0.000%	Off
8 ⚠	VPC Flow Agg to ...	`cribl_breaker=='AWS VPC Flow'`	agg	0.000%	Off
9-10	groupfoo			0.000%	Off
11 ⚠	PAN1 Data	`sourcetype=='pan1'`	devnull devnull:devnull	0.000%	Off
12 ⚠	Convert metrics.l...	`source.endsWith('metrics.log') ...`	logs2metrics	0.000%	Off
13 ⚠	Internal to localS...	`index.startsWith('_')`	bro devnull:devnull	0.000%	Off
14 ⚠	panRoute	`true`	palo_alto_traffic	0.000%	Off
15 ⚠	outViaCEFsdfsda	`sourcetype=='cefOUT'`	cefTest	0.000%	Off
16 ⚠	drop50%	`Math.random()>0.8`	devnull devnull:devnull	15.963%	On
17 ⚠	Internal metrics t...	`index=='_metrics'`	passthru splunk:localSplunk	65.342%	On

Unreachable Route warnings, many

Routing with Output Router

Output Router Destinations offer another way to route data. These function as meta-Destinations, in that they allow you to send data to multiple peer Destinations based on rules. Rules are evaluated in order, top->down, with the first match being the winner.

;

5.4. 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.

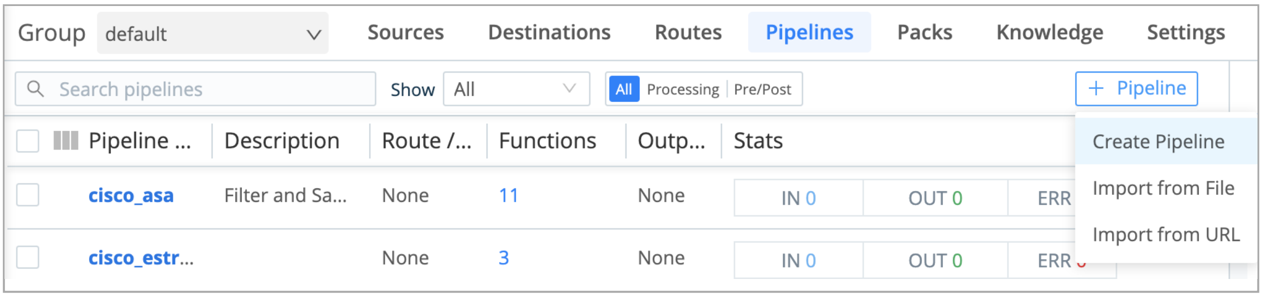
Accessing Pipelines

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.

Adding Pipelines

To create a new Pipeline, or to import an existing Pipeline to a different Cribl Stream instance, click **+ 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 ...	Description	Route /...	Functions	Outp...	Stats
cisco_asa	Filter and Sa...	None	11	None	IN 0 OUT 0 ERR
cisco_estr...		None	3	None	IN 0 OUT 0 ERR

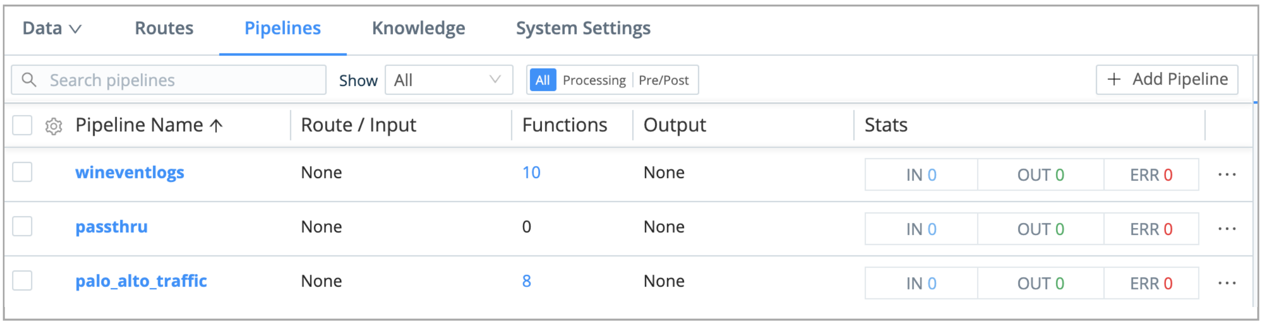
Creating or importing a Pipeline

> To export a Pipeline, see Advanced Mode (JSON Editor).

> To import or export a Pipeline along with broader infrastructure (like Knowledge Objects and/or sample data files), see Packs.

How Do Pipelines Work

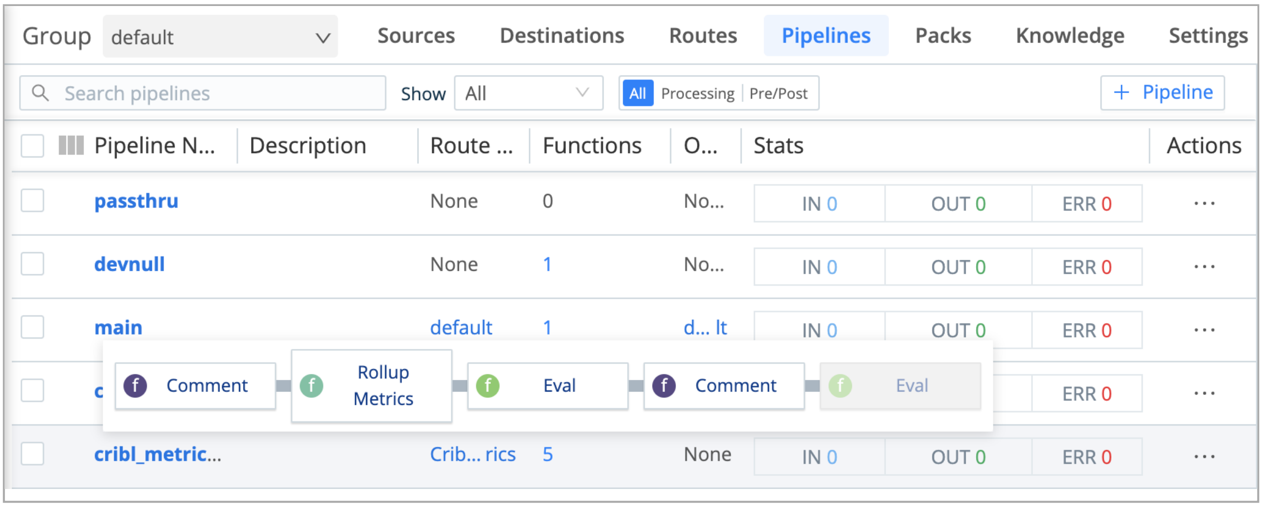
Events are always delivered to the beginning of a Pipeline via a Route. The data in the **Stats** column shown below are for the last 15 minutes.



Pipeline Name ↑	Route / Input	Functions	Output	Stats			
wineventlogs	None	10	None	IN 0	OUT 0	ERR 0	···
passthru	None	0	None	IN 0	OUT 0	ERR 0	···
palo_alto_traffic	None	8	None	IN 0	OUT 0	ERR 0	···

Pipelines and Route inputs

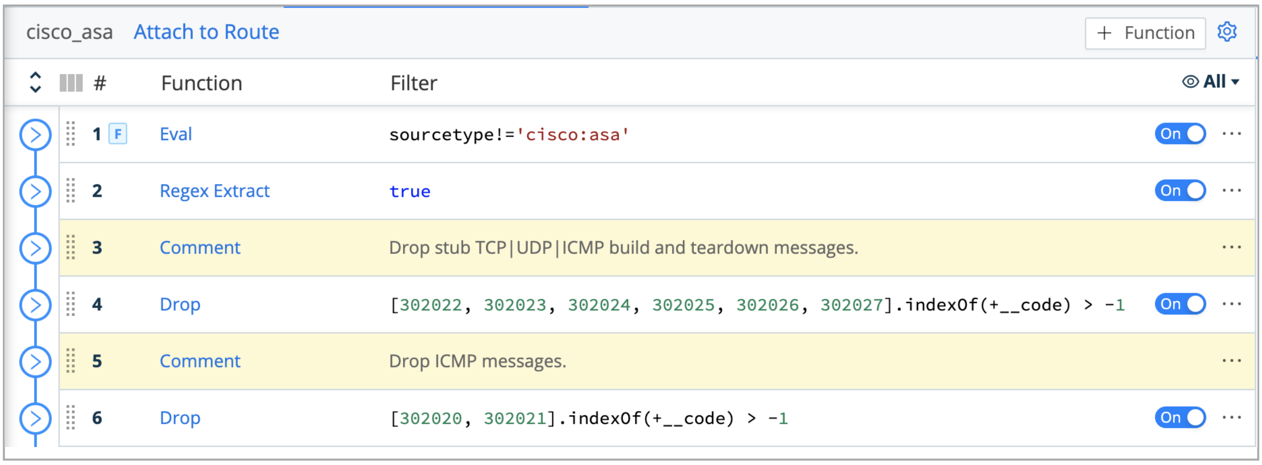
> You can press the] (right-bracket) shortcut key to toggle between the Preview pane and an expanded Pipelines display. (This shortcut works when no field has focus.)
>
> In the condensed Pipelines display above, you can also hover over any Pipeline's **Functions** column to see a horizontal preview of the stack of Functions contained in the Pipeline:



Pipeline N...	Description	Route ...	Functions	O...	Stats			Actions
passthru		None	0	No...	IN 0	OUT 0	ERR 0	···
devnull		None	1	No...	IN 0	OUT 0	ERR 0	···
main		default	1	d... lt	IN 0	OUT 0	ERR 0	···
c							ERR 0	···
cribl_metric...		Crib... rics	5	None	IN 0	OUT 0	ERR 0	···

Preview on hovering over the bottom Pipeline (highlighted in gray)

Within the Pipeline, events are processed by each Function, in order. A Pipeline will always move events in the direction that points outside of the system. This is on purpose, to keep the design simple and avoid potential loops.



cisco_asa Attach to Route

#	Function	Filter	
1 F	Eval	`sourcetype!='cisco:asa'`	On
2	Regex Extract	`true`	On
3	Comment	Drop stub TCP|UDP|ICMP build and teardown messages.	
4	Drop	`[302022, 302023, 302024, 302025, 302026, 302027].indexOf(+__code) > -1`	On
5	Comment	Drop ICMP messages.	
6	Drop	`[302020, 302021].indexOf(+__code) > -1`	On

Pipeline Functions

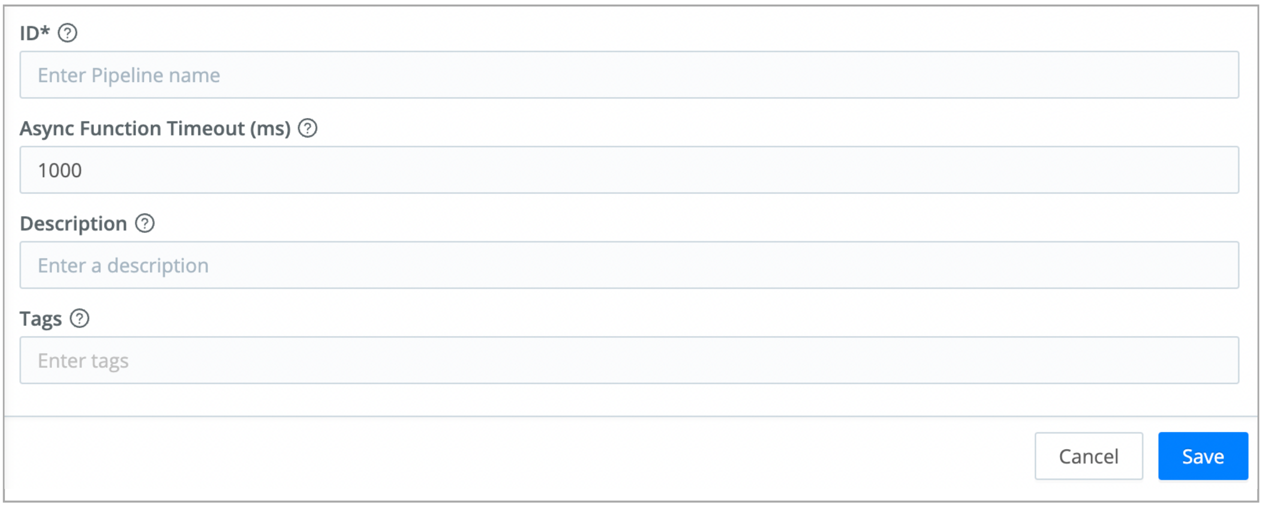
> Use the **Attach to Route** link at upper left to associate a new Pipeline with a Route.
>
> You can streamline a complex Pipeline's display by organizing related Functions into Function groups.

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

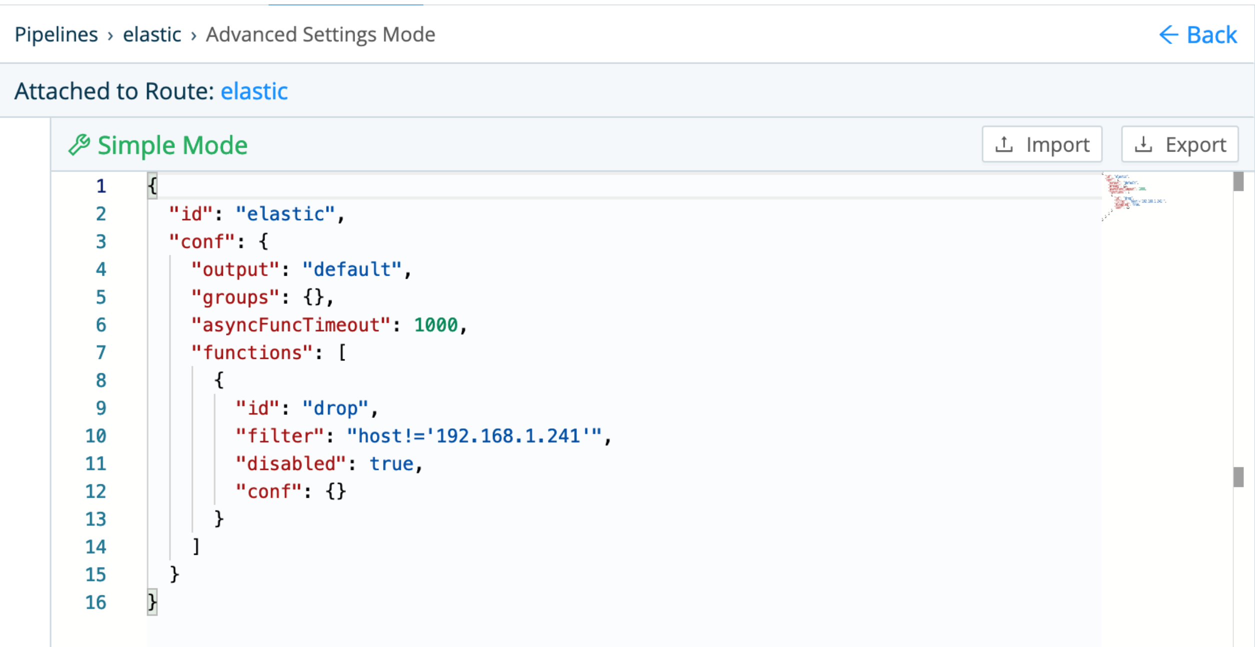
Tags ?

Enter tags

Pipeline Settings

Advanced Mode (JSON Editor)

Once you've clicked the gear button to enter Pipeline Settings, you can click **Edit as JSON** at the upper right to edit the Pipeline's definition in a JSON text editor. In this mode's editor, you can directly edit multiple values. You can also use the **Import** and **Export** buttons here to copy and modify existing Pipeline configurations, as `.json` files.



Pipelines › elastic › Advanced Settings Mode

Attached to Route: elastic

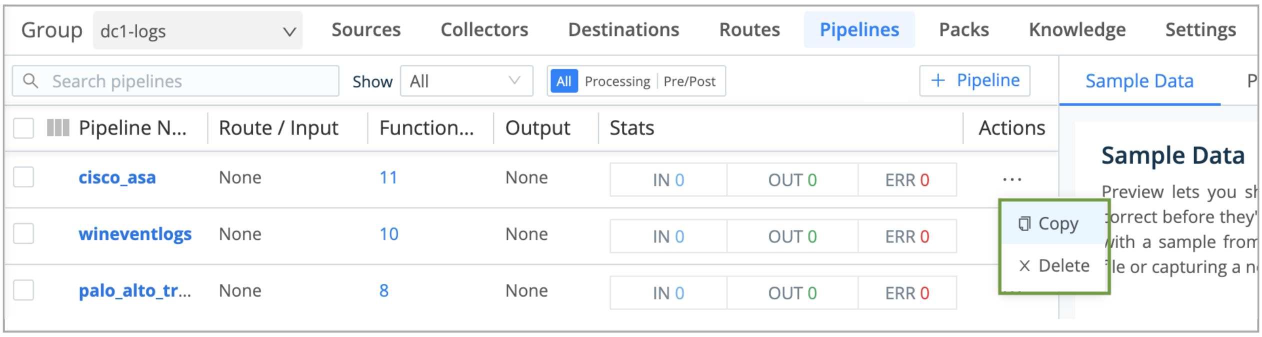
```
{
  "id": "elastic",
  "conf": {
    "output": "default",
    "groups": {},
    "asyncFuncTimeout": 1000,
    "functions": [
      {
        "id": "drop",
        "filter": "host!='192.168.1.241'",
        "disabled": true,
        "conf": {}
      }
    ]
  }
}
```

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

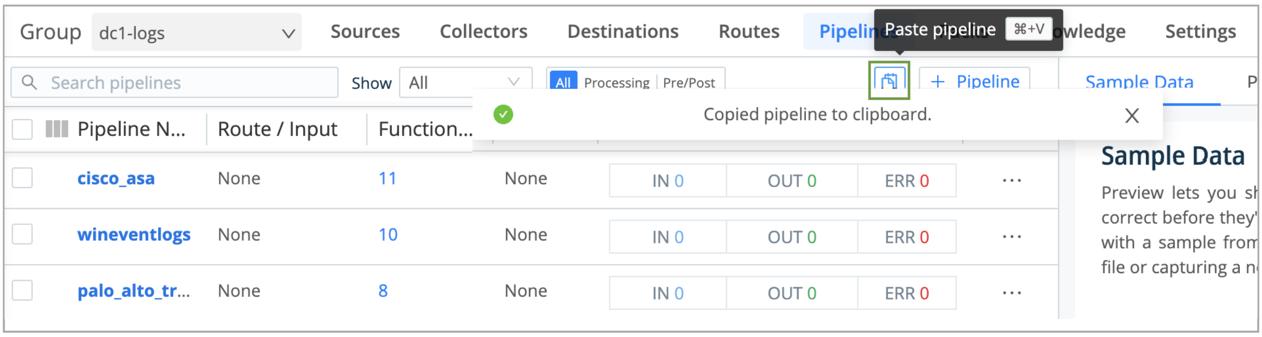
Click a Pipeline's Actions (**...**) menu to display options for copying or deleting the Pipeline.



Pipeline N...	Route / Input	Function...	Output	Stats			Actions
cisco_asa	None	11	None	IN 0	OUT 0	ERR 0	...
wineventlogs	None	10	None	IN 0	OUT 0	ERR 0	
palo_alto_tr...	None	8	None	IN 0	OUT 0	ERR 0	

Pipeline > Actions menu

Copying a Pipeline displays the confirmation message and the (highlighted) Paste button shown below.

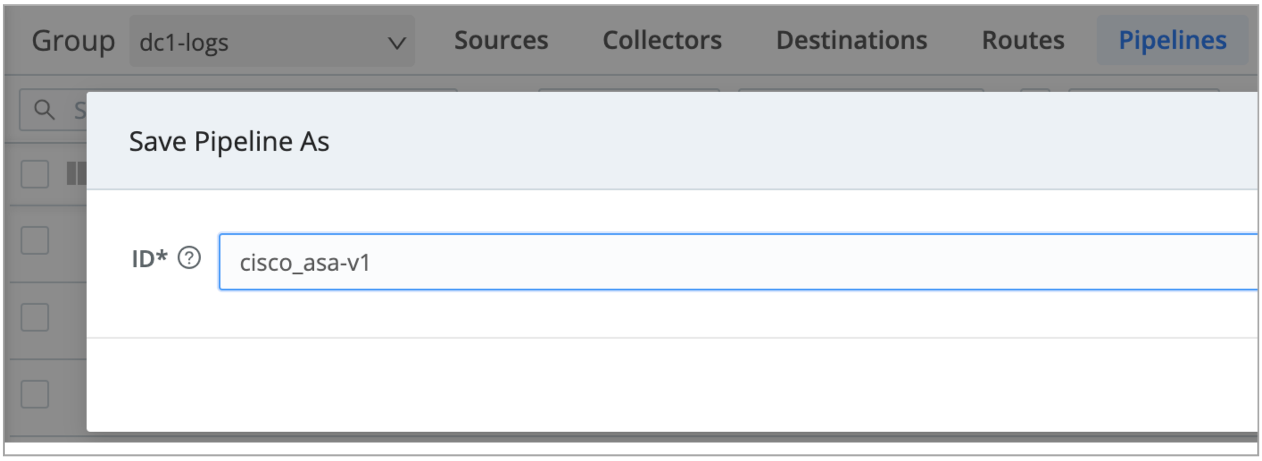


Copied pipeline to clipboard.

Pipeline N...	Route / Input	Function...					
cisco_asa	None	11	None	IN 0	OUT 0	ERR 0	...
wineventlogs	None	10	None	IN 0	OUT 0	ERR 0	...
palo_alto_tr...	None	8	None	IN 0	OUT 0	ERR 0	...

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.



Save Pipeline As

ID* cisco_asa-v1

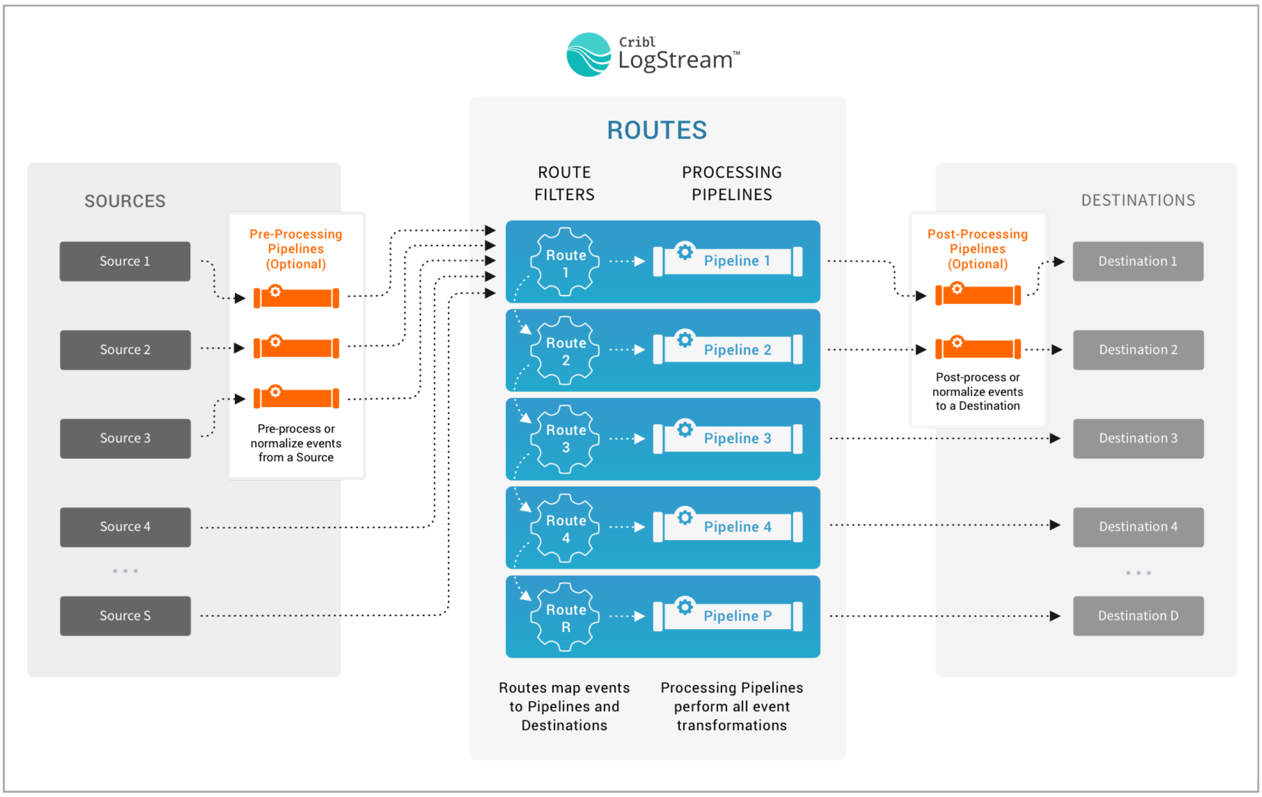
Saving/renaming a pasted Pipeline

Chaining Pipelines

In Cribl Stream (LogStream) 3.2.x and above, you can use the Chain Function to send the output of a Pipeline to another Pipeline or Pack. There are scope restrictions within Packs, and general guardrails against circular references.

Types of Pipelines

You can apply various Pipeline types at different stages of data flow. All Pipelines have the same basic internal structure (a series of Functions) – the types below differ only in their position in the system.



Cribl LogStream™
ROUTES
ROUTE FILTERS
PROCESSING PIPELINES
SOURCES
Source 1
Source 2
Source 3
Source 4
Source S
Pre-Processing Pipelines (Optional)
Pre-process or normalize events from a Source
Route 1
Route 2
Route 3
Route 4
Route R
Pipeline 1
Pipeline 2
Pipeline 3
Pipeline 4
Pipeline P
Routes map events to Pipelines and Destinations
Processing Pipelines perform all event transformations
Post-Processing Pipelines (Optional)
Post-process or normalize events to a Destination
DESTINATIONS
Destination 1
Destination 2
Destination 3
Destination 4
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.

;

5.5. 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. You can develop and test a configuration, and then port it from development to production instances, or readily deploy it to multiple Worker Groups/Fleets.

We don't claim to have brokered world peace here, but we do modestly hope to promote a stable, prosperous Pax Criblatica for the Cribl Stream ecosystem.

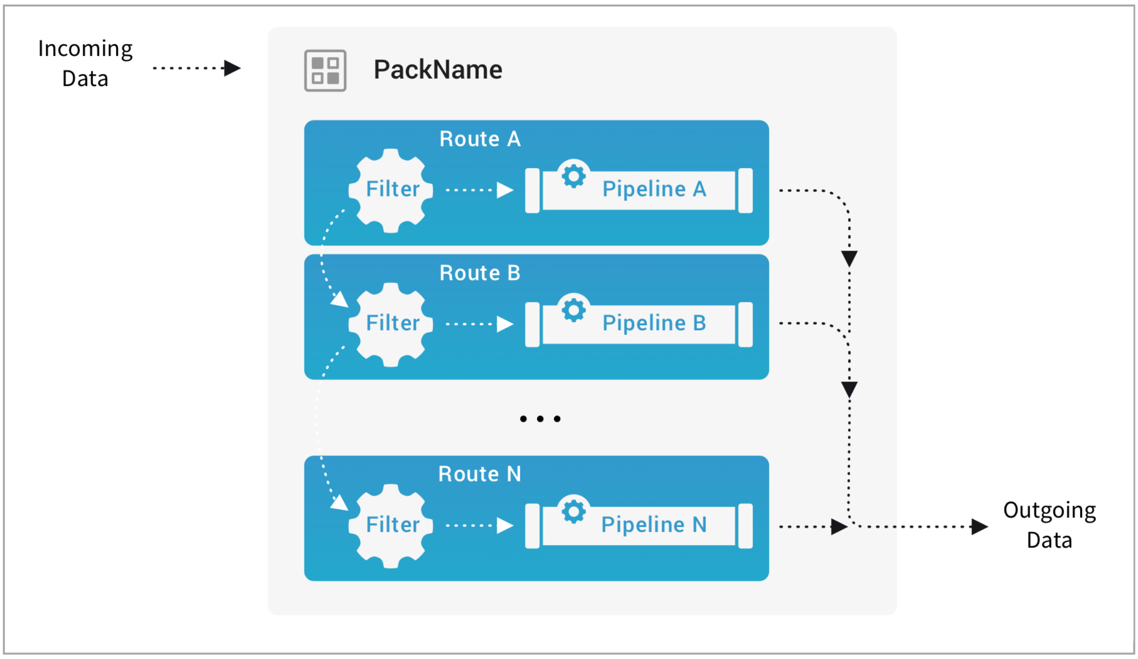
What Is a Pack?

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.

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

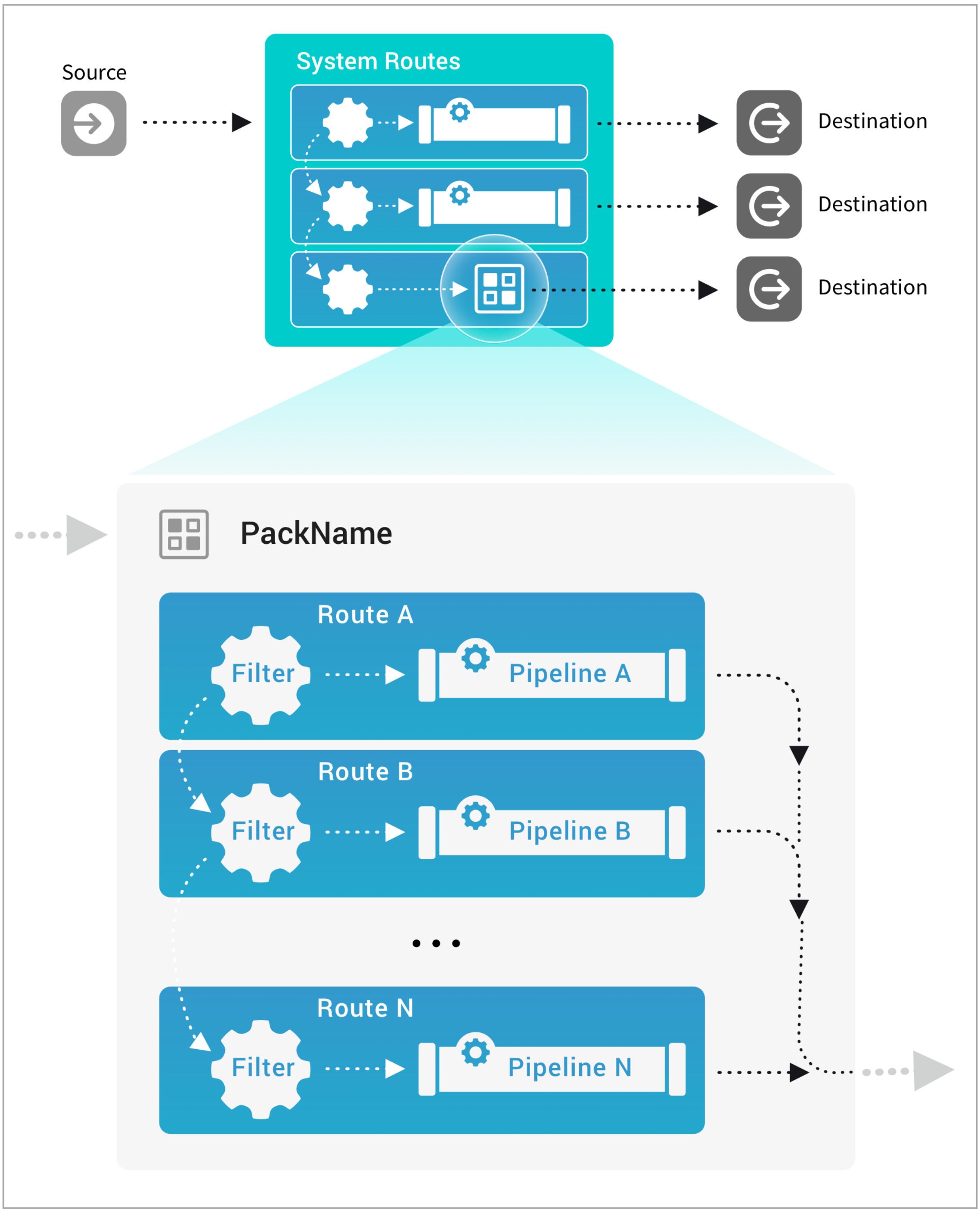
These instructions cover using predefined Packs, as well as creating and modifying Pack configurations.

Where Can I Use Packs?

Wherever you can reference a Pipeline, you can specify a Pack:

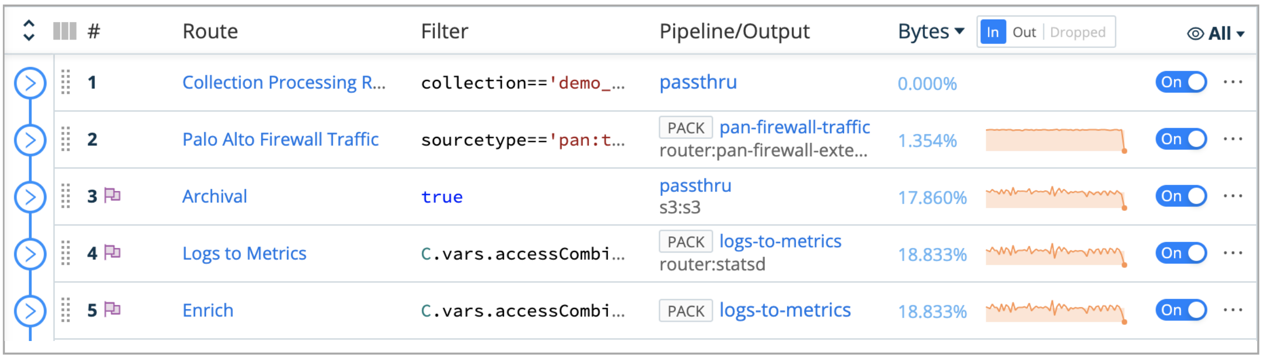
- In Sources, where you attach pre-processing Pipelines.
- In Destinations, where you attach post-processing Pipelines.
- In Routes, in the Routing table's **Pipeline/Output** column.

This expanded view shows how a Pack can replace a Pipeline in a Route:



A Pack snaps into Cribl Stream like an enhanced Pipeline

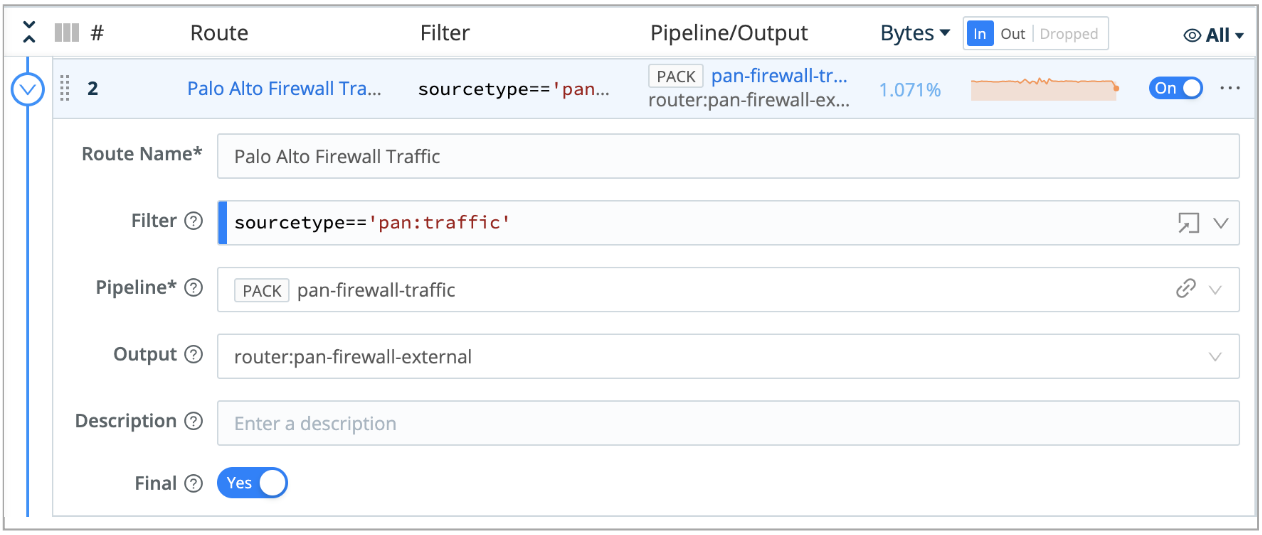
Packs are distinguished in the display with a **PACK** badge, as you can see here in the Routing table:



#	Route	Filter	Pipeline/Output	Bytes	In Out Dropped	All
1	Collection Processing R...	`collection=='demo_...`	passthru	0.000%		On
2	Palo Alto Firewall Traffic	`sourcetype=='pan:t...`	PACK pan-firewall-traffic router:pan-firewall-exte...	1.354%		On
3	Archival	`true`	passthru s3:s3	17.860%		On
4	Logs to Metrics	`C.vars.accessCombi...`	PACK logs-to-metrics router:statsd	18.833%		On
5	Enrich	`C.vars.accessCombi...`	PACK logs-to-metrics	18.833%		On

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:



#	Route	Filter	Pipeline/Output	Bytes	In Out Dropped	All
2	Palo Alto Firewall Tra...	`sourcetype=='pan...`	PACK pan-firewall-tr... router:pan-firewall-ex...	1.071%		On

Route Name*: Palo Alto Firewall Traffic

Filter: `sourcetype=='pan:traffic'`

Pipeline*: PACK pan-firewall-traffic

Output: router:pan-firewall-external

Description: Enter a description

Final: Yes

PACK badge on a Pack connected to a Route

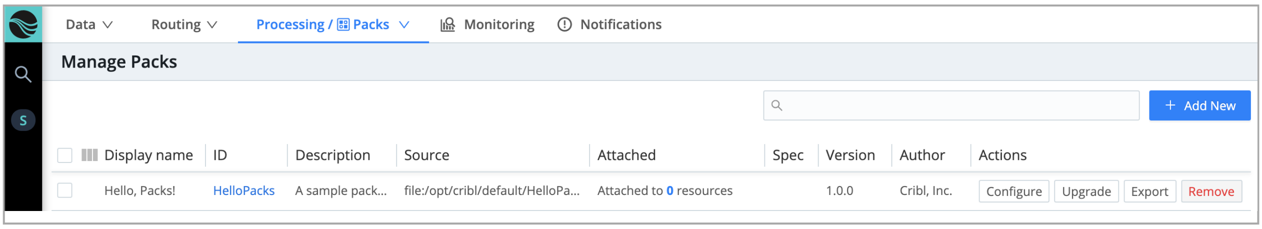
Cribl Stream's **Monitoring** page includes a **Packs** link where you can monitor Packs' throughput.

Accessing Packs

You access Packs differently, depending on your deployment type.

Single-Instance

In a single-instance deployment, Packs are global. From Cribl Stream's top-level navigation, just select **Processing** > **Packs**.



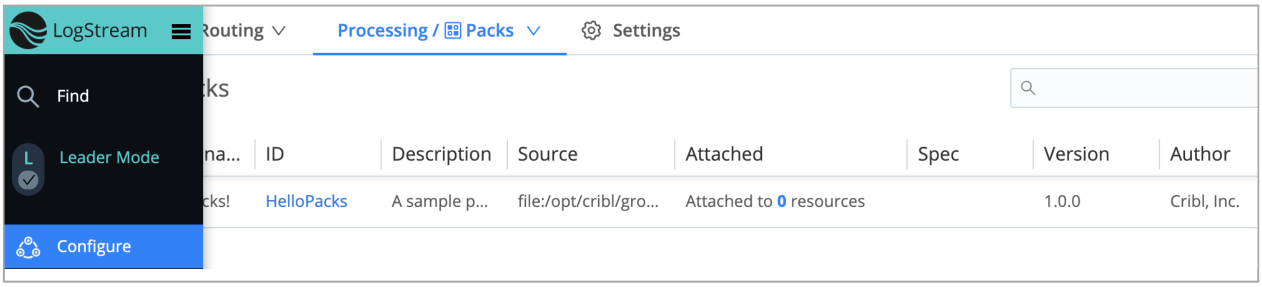
Manage Packs

Display name	ID	Description	Source	Attached	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack...	file:/opt/cribl/default/HelloPa...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove

Packs, single-instance navigation

Distributed/Default Worker Group/Fleet

In a distributed deployment with the default single Worker Group/Fleet (Leader mode), select **Configure** from the left nav, then **Processing** > **Packs** from the resulting top nav.



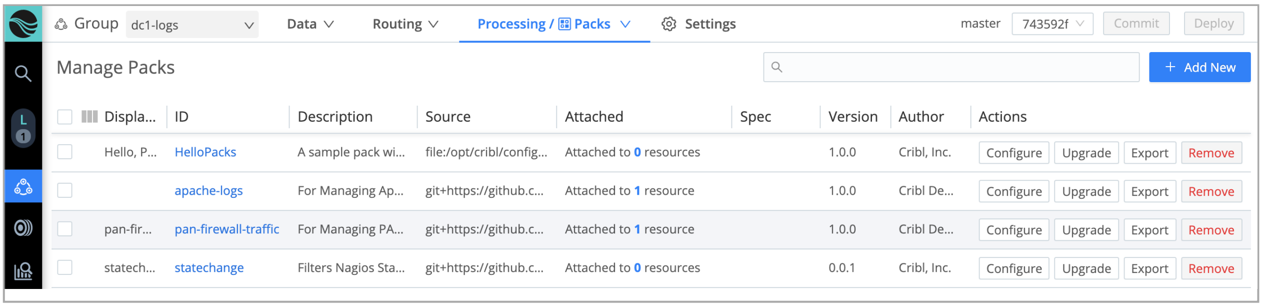
ks

na...	ID	Description	Source	Attached	Spec	Version	Author
cks!	HelloPacks	A sample p...	file:/opt/cribl/gro...	Attached to 0 resources		1.0.0	Cribl, Inc.

Packs, Leader mode

Distributed/Multiple Worker Groups/Fleets

In a distributed deployment with multiple Worker Groups/Fleets (Leader mode), Packs are associated with (and installed within) Worker Groups. Navigate to the parent Worker Group/Fleet, then select **Processing** > **Packs** from that Group's top nav.



Manage Packs

Displa...	ID	Description	Source	Attached	Spec	Version	Author	Actions
Hello, P...	HelloPacks	A sample pack wi...	file:/opt/cribl/config...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
	apache-logs	For Managing Ap...	git+https://github.c...	Attached to 1 resource		1.0.0	Cribl De...	Configure Upgrade Export Remove
pan-fir...	pan-firewall-traffic	For Managing PA...	git+https://github.c...	Attached to 1 resource		1.0.0	Cribl De...	Configure Upgrade Export Remove
statech...	statechange	Filters Nagios Sta...	git+https://github.c...	Attached to 0 resources		0.0.1	Cribl, Inc.	Configure Upgrade Export Remove

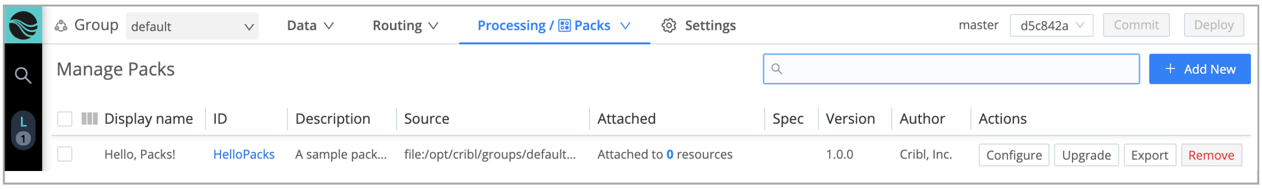
Worker Group/Fleet > Manage Packs page

> As the top nav adds more controls on narrower browsers, **Packs** and other right-side links can move onto the ••• overflow menu, as shown above.

> By design, you can readily share Packs **across** Worker Groups/Fleets by exporting/importing them (both covered below).

Getting Started with Packs

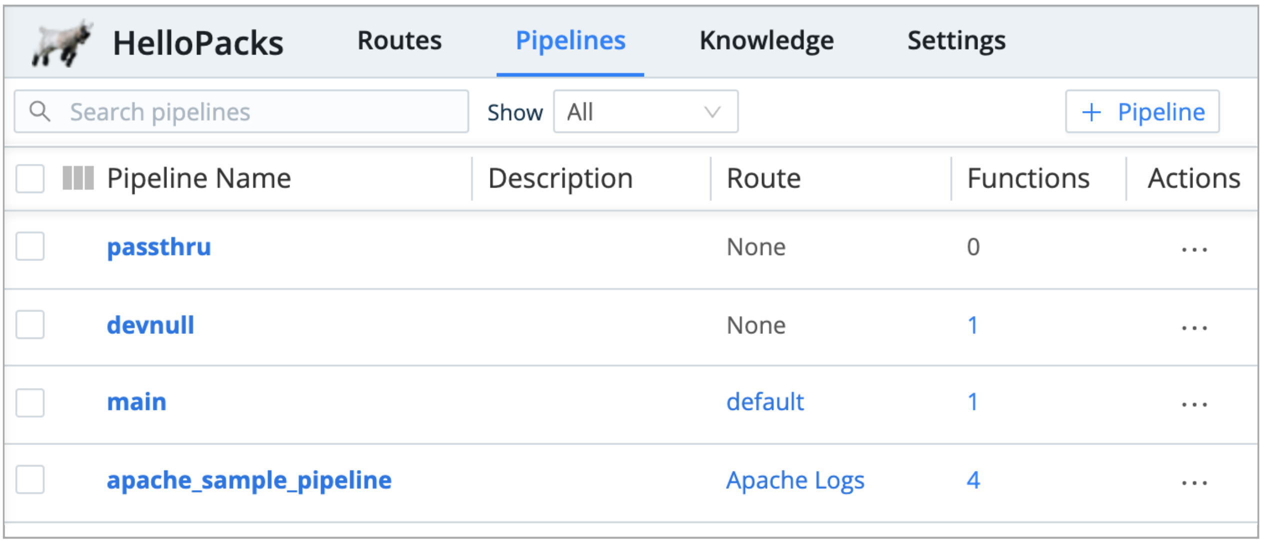
To unpack Packs, use the above instructions (per deployment type) to navigate to the **HelloPacks** example Pack shipped with Cribl Stream. On the **Manage Packs** page, click this Pack's row to see its configuration.



Display name	ID	Description	Source	Attached	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack...	file:/opt/cribl/groups/default...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove

Manage Packs page with example Pack

Click **Pipelines** on the Pack's submenu, and you'll see that the Pack includes `devnull`, `main`, and `passthru` Pipelines, corresponding to the default Pipelines provided at Cribl Stream's global level. This Pack also includes an Apache-specific sample Pipeline – click it to unpack that, too.

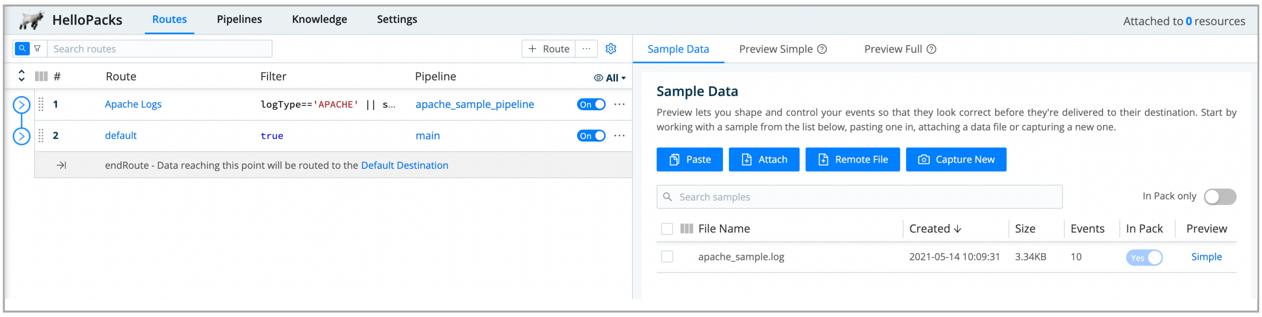


Pipeline Name	Description	Route	Functions	Actions
passthru		None	0	...
devnull		None	1	...
main		default	1	...
apache_sample_pipeline		Apache Logs	4	...

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 – a subset of Cribl Stream's top nav – above it: **Routes**, **Pipelines**, **Knowledge**, and **Settings** on the left pane, along with **Sample Data**, and **Preview Simple** on the right.



#	Route	Filter	Pipeline	
1	Apache Logs	`logType=='APACHE' || s...`	apache_sample_pipeline	On
2	default	`true`	main	On

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

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.

File Name	Created ↓	Size	Events	In Pack	Preview
apache_sample.log	2021-05-14 10:09:31	3.34KB	10	Yes	Simple

Configuring a Pack

The left pane's links give you access to configuration objects specific to this Pack.

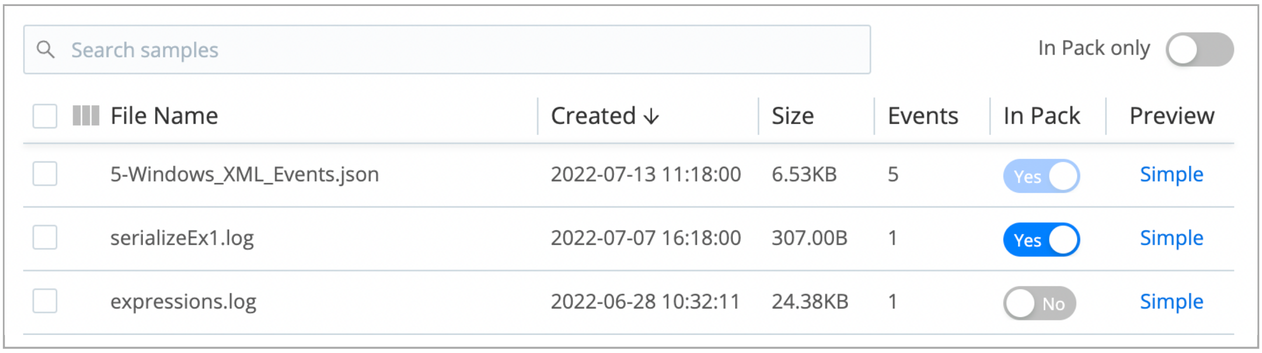
Sample Data

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 the **In Pack only** slider to the right.

If you add sample data files via this Pack UI, they will be internal to that Pack. Each sample file here displays its own **In Pack** toggle on its row, which works as follows:

A light-blue toggle is locked, meaning that this sample file is internal to the Pack. It will export with the Pack. If you want to make this sample available across Cribl 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

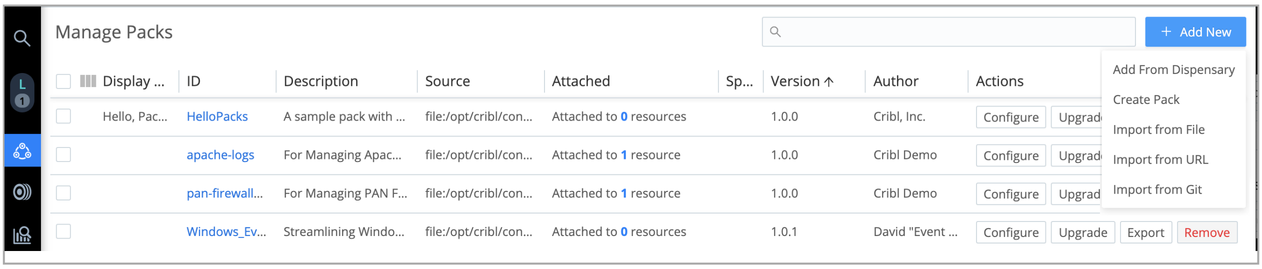
Sample file in Pack

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 New** 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.



Manage Packs

Display ...	ID	Description	Source	Attached	Sp...	Version ↑	Author	Actions
Hello, Pac...	HelloPacks	A sample pack with ...	file:/opt/cribl/con...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrad
	apache-logs	For Managing Apac...	file:/opt/cribl/con...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrad
	pan-firewall...	For Managing PAN F...	file:/opt/cribl/con...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrad
	Windows_Ev...	Streamlining Windo...	file:/opt/cribl/con...	Attached to 0 resources		1.0.1	David "Event ...	Configure Upgrade Export Remove

Importing a Pack

Custom Functions

Packs can include Pipelines containing custom functions, which can (in turn) run arbitrary JavaScript. Before you install a Pack, make sure it comes from a provider you trust, such as the Cribl Packs Dispensary or your own organization.

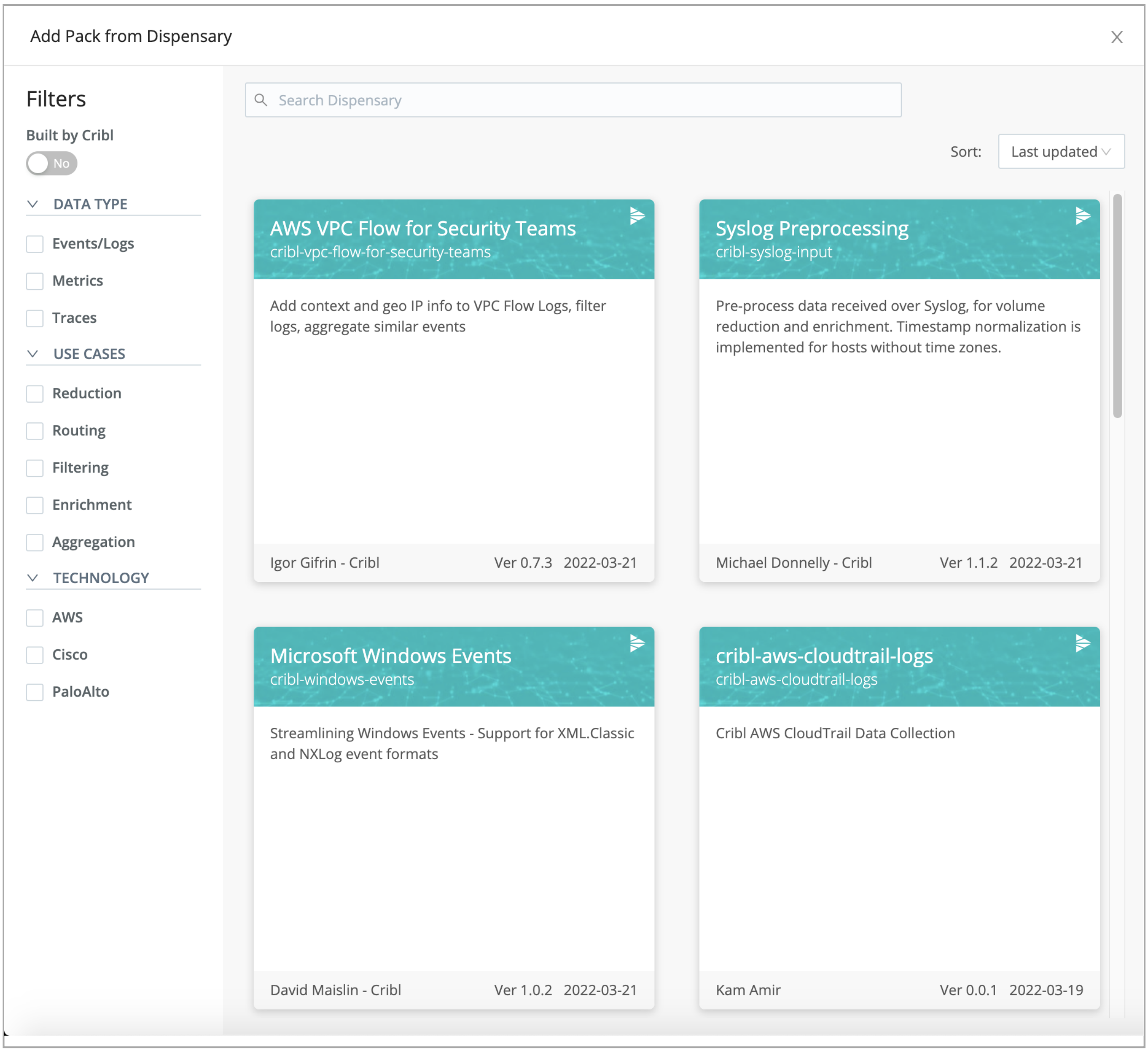
As an additional protection layer, all Pack import modals provide an **Allow custom functions** slider. In the slider's default `No` position, if Cribl Stream detects custom functions in the specified Pack, it will block the import with an error message. If you trust the Pack's provider, toggle the slider to `Yes`, and the import will proceed normally.

The Cribl Packs Dispensary™

You might be wondering, "Where can I find a reliable source of Packs that add useful features to Cribl 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 this repo 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)

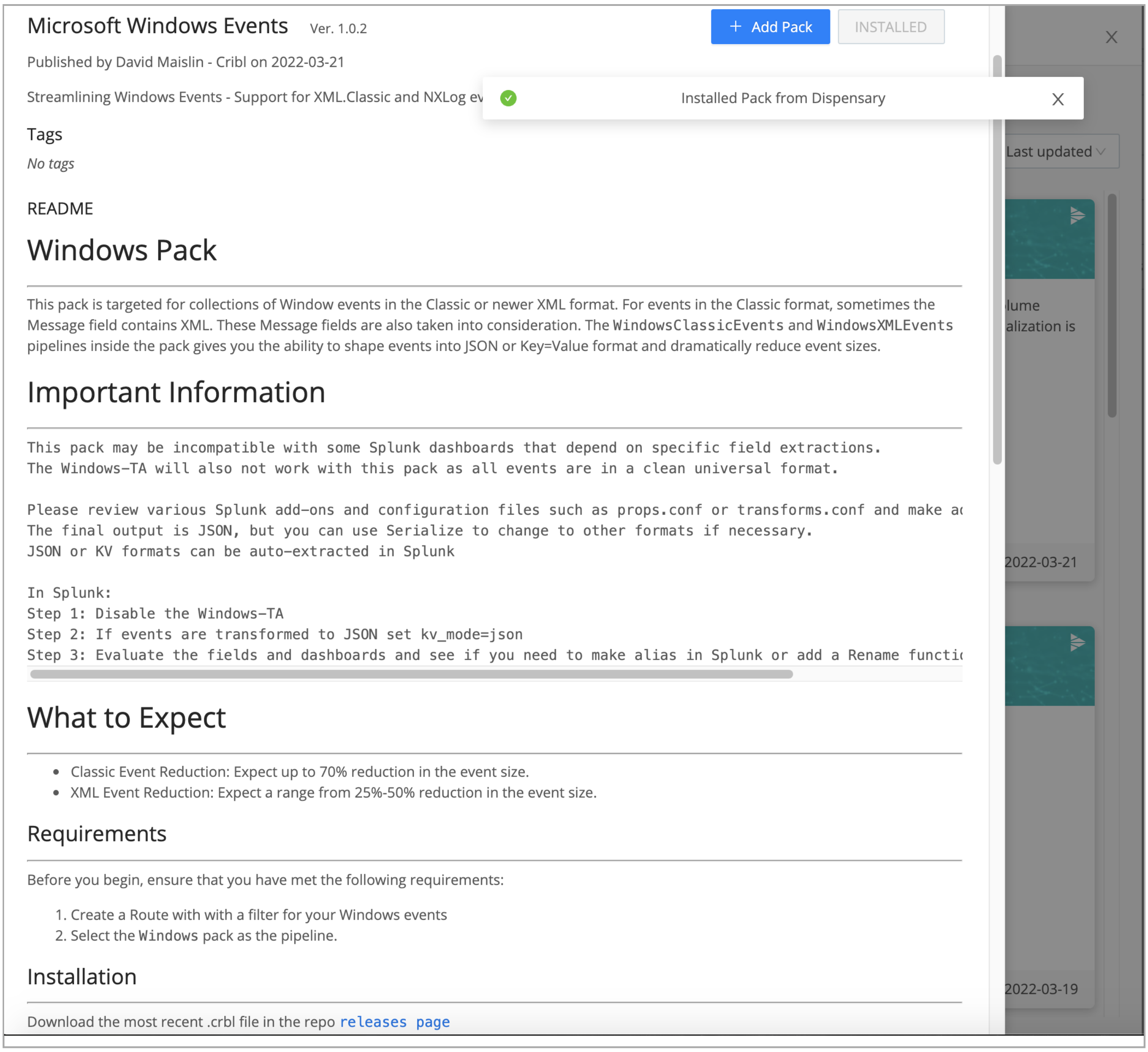
> Interested in publishing your own Packs on the Cribl Packs Dispensary™? See Publishing a Pack.

Add from Packs Dispensary

To add a Pack from the Cribl Packs Dispensary™ sharing site:

1. From the **Manage Packs** page's **+ Add New** 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/technology and purpose.)
4. Click any Pack's tile to display its details page. 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.



Microsoft Windows Events Ver. 1.0.2

Published by David Maislin - Cribl on 2022-03-21

Streamlining Windows Events - Support for XML.Classic and NXLog ev

Installed Pack from Dispensary

Tags

No tags

README

Windows Pack

This pack is targeted for collections of Window events in the Classic or newer XML format. For events in the Classic format, sometimes the Message field contains XML. These Message fields are also taken into consideration. The `WindowsClassicEvents` and `WindowsXMLEvents` pipelines inside the pack gives you the ability to shape events into JSON or Key=Value format and dramatically reduce event sizes.

Important Information

```
This pack may be incompatible with some Splunk dashboards that depend on specific field extractions.
The Windows-TA will also not work with this pack as all events are in a clean universal format.

Please review various Splunk add-ons and configuration files such as props.conf or transforms.conf and make ac
The final output is JSON, but you can use Serialize to change to other formats if necessary.
JSON or KV formats can be auto-extracted in Splunk

In Splunk:
Step 1: Disable the Windows-TA
Step 2: If events are transformed to JSON set kv_mode=json
Step 3: Evaluate the fields and dashboards and see if you need to make alias in Splunk or add a Rename functio
```

What to Expect

- Classic Event Reduction: Expect up to 70% reduction in the event size.
- XML Event Reduction: Expect a range from 25%-50% reduction in the event size.

Requirements

Before you begin, ensure that you have met the following requirements:

1. Create a Route with with a filter for your Windows events
2. Select the `Windows` pack as the pipeline.

Installation

Download the most recent .crbl file in the repo `releases page`

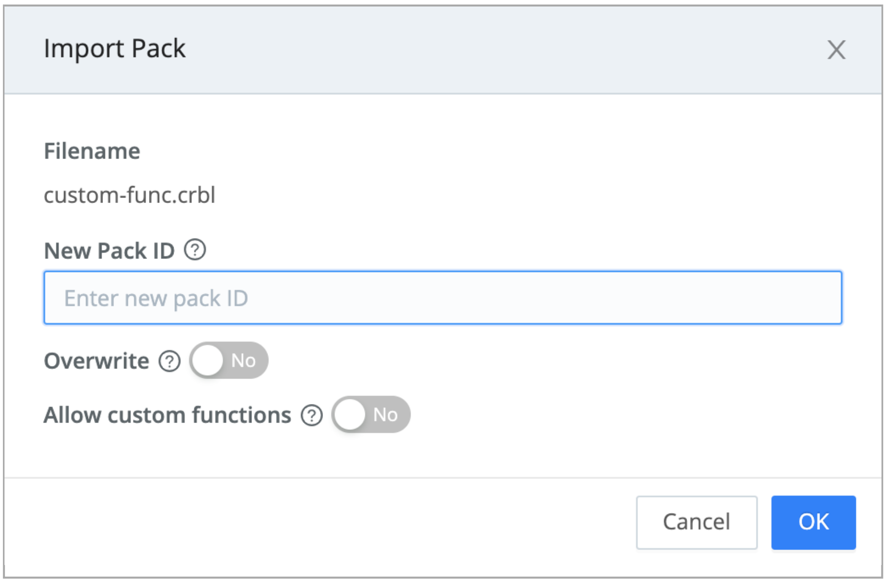
Pack details page: Composite with **+ Add Pack** button, confirmation banner, Dispensary drawer in background

Import from File

To import a Pack (`.crbl` file) from your local filesystem:

1. From the **+ Add New** 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

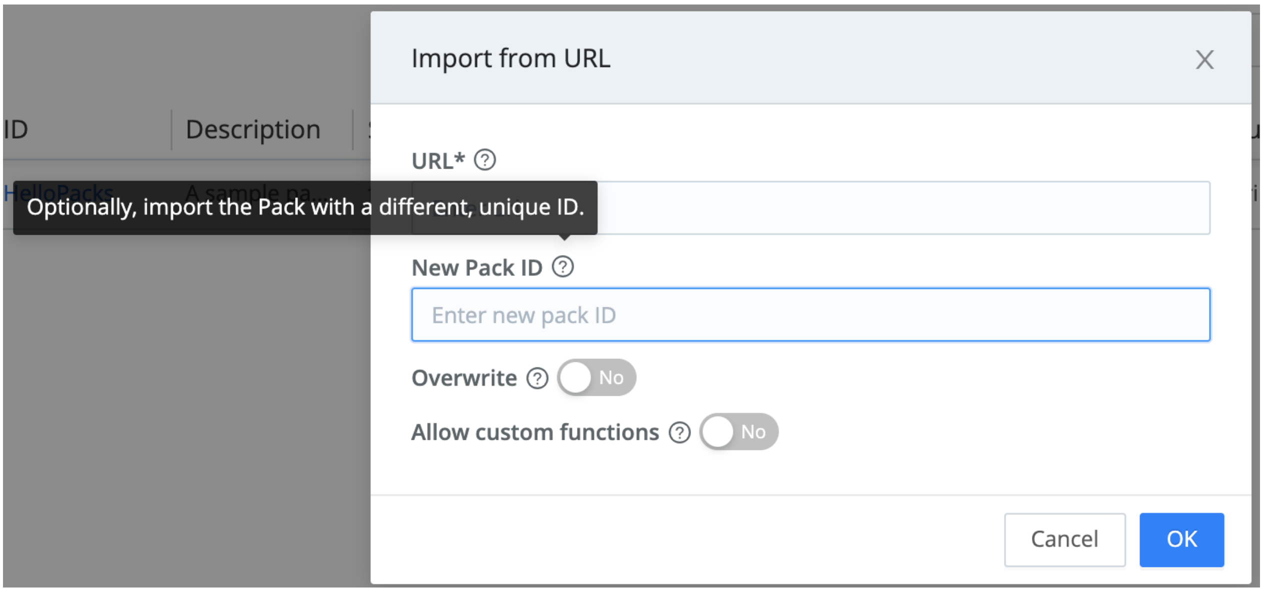
Cancel OK

Importing from a file

Import from URL

To import a Pack from a known, public or internal, URL:

1. From the **+ Add New** 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.



ID | Description

HelloPacks | A sample pa

Optionally, import the Pack with a different, unique ID.

Import from URL

URL*

New Pack ID

Enter new pack ID

Overwrite: No

Allow custom functions: No

Cancel OK

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 **+ Add New** submenu, select **Import from Git**.

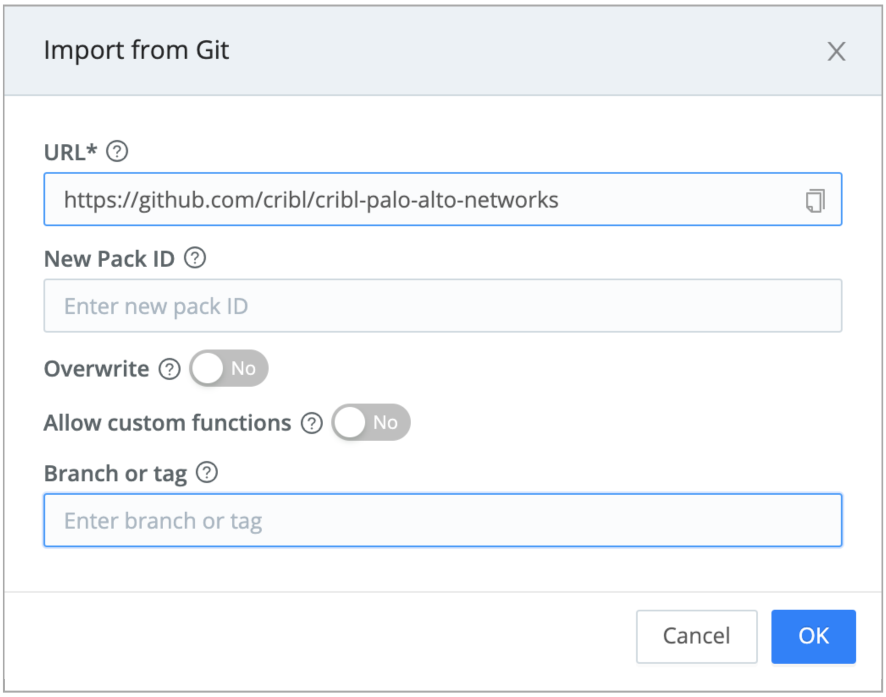
2. Enter the source repo's valid URL.
   This field's input is validated for URL format, but not for completeness or accuracy, before you submit the modal. When targeting a private repo, use the format: `https://<username>:<token/password>:<repo-address>`. Public repos need only `https://<repo-address>`, 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

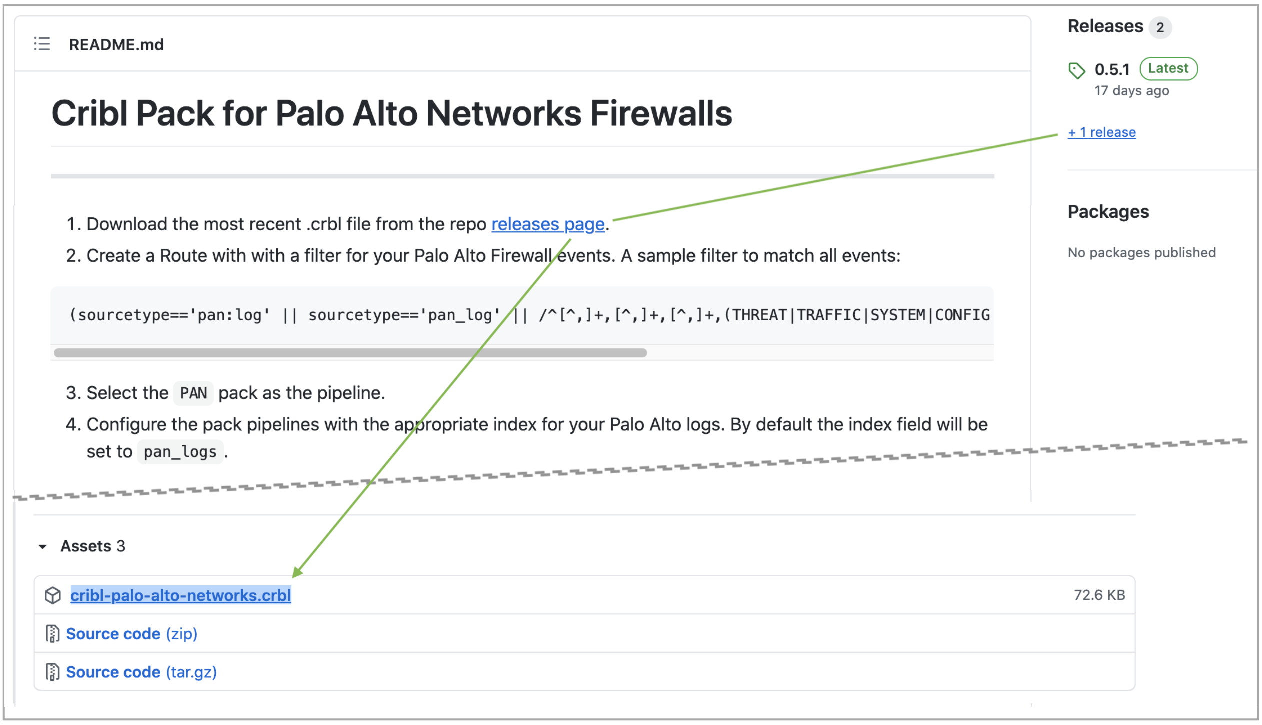
Importing from a Git repo

> 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` .

Assets 3

cribl-palo-alto-networks.crbl	72.6 KB
Source code (zip)	
Source code (tar.gz)	

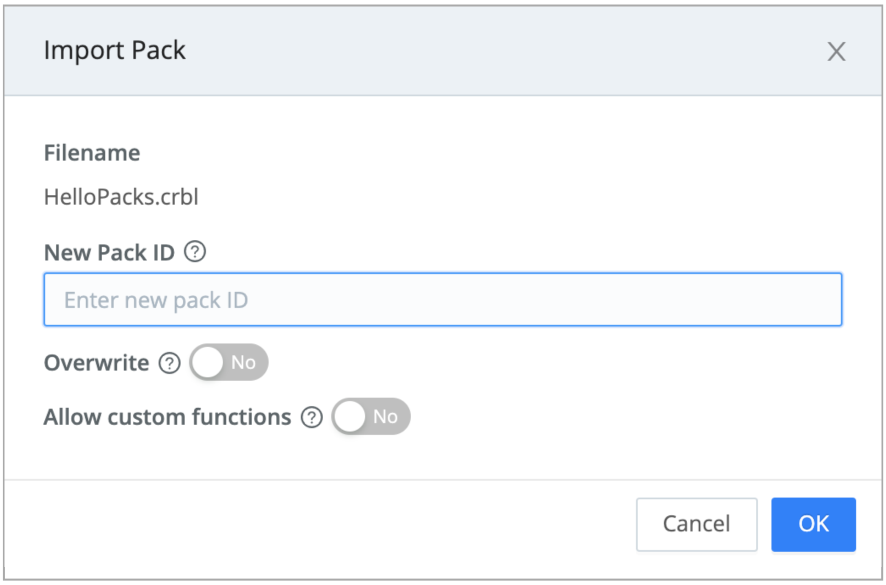
Downloading a `.crbl` file from the Cribl Pack Dispensary's Web UI

> 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/Fleet (or single-instance deployment) must have a unique ID. The ID is based on the Pack's internal configuration – not its container's file name, nor on its Display name.

If you import a Pack whose internal ID matches an installed Pack – whether an update, or just a duplicate – you'll be prompted to assign a unique **New Pack ID** to import it as a separate Pack.



Import Pack

Filename

HelloPacks.crbl

New Pack ID

Enter new pack ID

Overwrite: No

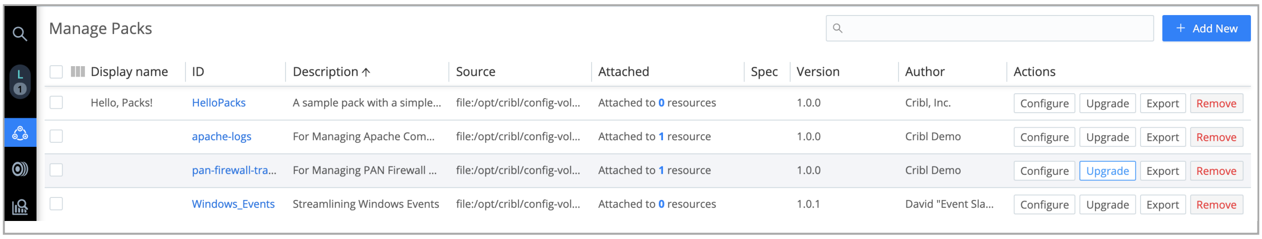
Allow custom functions: No

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 **Upgrade** button on its row.



Manage Packs

□	Display name	ID	Description ↑	Source	Attached	Spec	Version	Author	Actions
□	Hello, Packs!	HelloPacks	A sample pack with a simple...	file:/opt/cribl/config-vol...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
□		apache-logs	For Managing Apache Com...	file:/opt/cribl/config-vol...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
□		pan-firewall-tra...	For Managing PAN Firewall ...	file:/opt/cribl/config-vol...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
□		Windows_Events	Streamlining Windows Events	file:/opt/cribl/config-vol...	Attached to 0 resources		1.0.1	David "Event Sla...	Configure Upgrade Export Remove

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.

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.

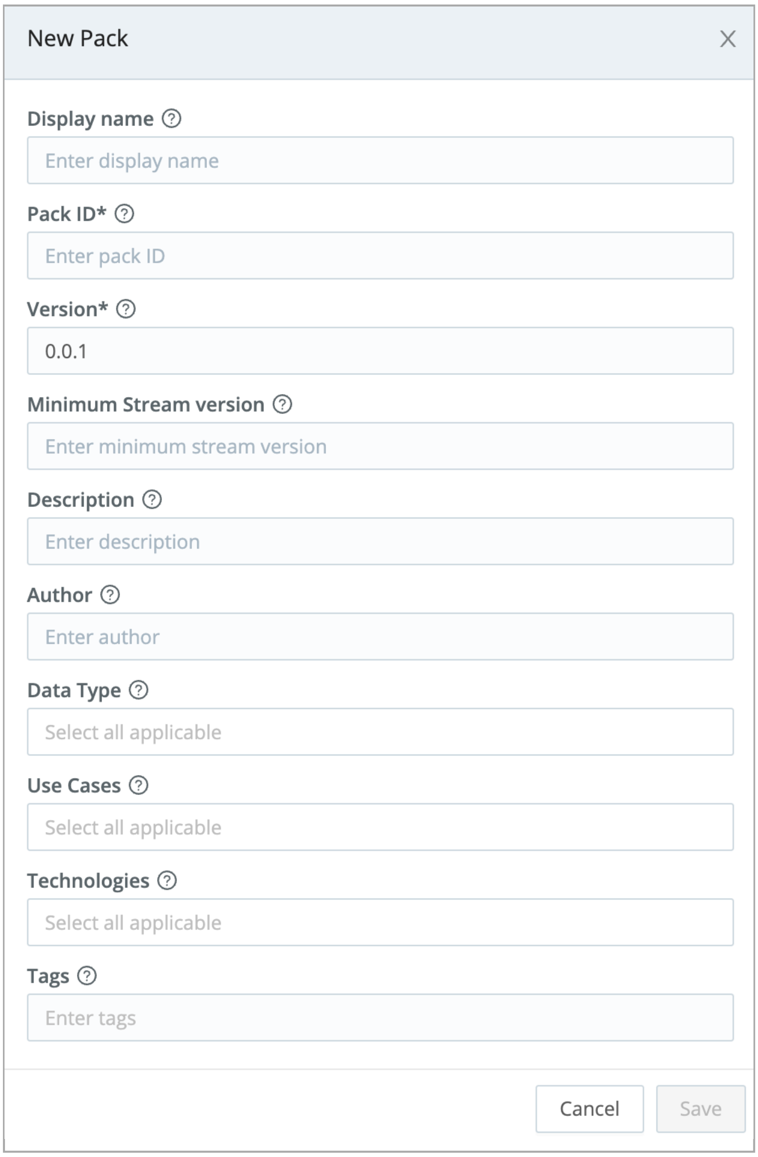
2. Click **+ Add New**.

3. From the submenu, select **Create Pack**.

4. In the resulting **New Pack** modal, fill in a unique **Pack ID** and other details.

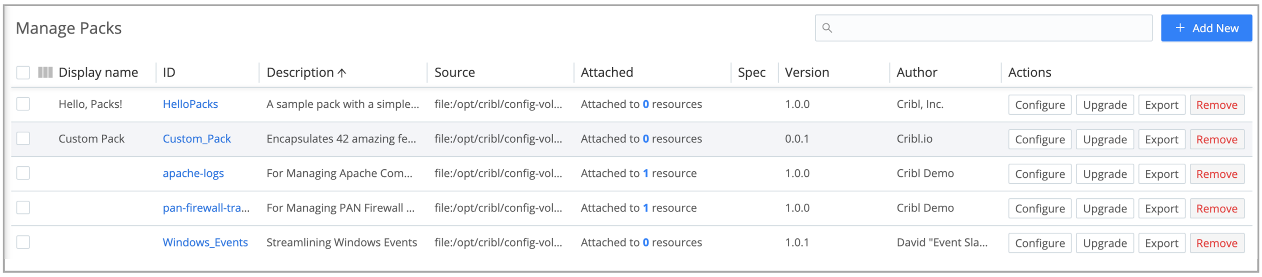
> - Each Pack within a Cribl Stream instance must have a separate **Pack ID**, but you can assign arbitrary **Display name**s.
> - **Version** is a required field identifying the Pack's own versioning.
> - **Minimum Stream version** is an optional field specifying the lowest compatible version of Cribl Stream/Edge software.
> - **Description** and **Author** are optional identifiers.
> - **Data type**, **Use cases**, and **Technologies** are optional combo boxes. You can insert one or multiple keywords to help users filter Packs that you post publicly on the Cribl Packs Dispensary™.
> - **Tags** are optional, arbitrary labels that you can use to filter/search and organize Packs.

5. Click **OK** to save the Pack.



Creating a Pack

6. On the **Manage Packs** page, click the new Pack's row to open the Pack.



□ Display name	ID	Description ↑	Source	Attached	Spec	Version	Author	Actions
□ Hello, Packs!	HelloPacks	A sample pack with a simple...	file:/opt/cribl/config-vol...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
□ Custom Pack	Custom_Pack	Encapsulates 42 amazing fe...	file:/opt/cribl/config-vol...	Attached to 0 resources		0.0.1	Cribl.io	Configure Upgrade Export Remove
□	apache-logs	For Managing Apache Com...	file:/opt/cribl/config-vol...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
□	pan-firewall-tra...	For Managing PAN Firewall ...	file:/opt/cribl/config-vol...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
□	Windows_Events	Streamlining Windows Events	file:/opt/cribl/config-vol...	Attached to 0 resources		1.0.1	David "Event Sla...	Configure Upgrade Export Remove

Manage Packs page

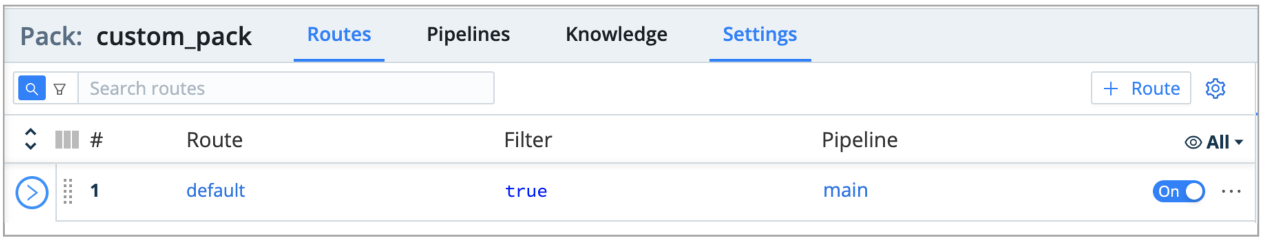
7. Use the standard Cribl Stream controls (see above) to configure and save the infrastructure you want to pack up. As you save changes in the UI, they're saved to the Pack.

> 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.
>
> Once your pack is ready to share, we encourage you to submit it to the Cribl Packs Dispensary™ site. If you already have completed Packs on our GitHub repo, bring them over here!

Modifying Pack Settings

You can update a Pack's metadata (Version, Description, Author, etc.) and display settings. If you're developing a new Pack to share, you'll want to use this interface to populate the Pack's README and display logo.

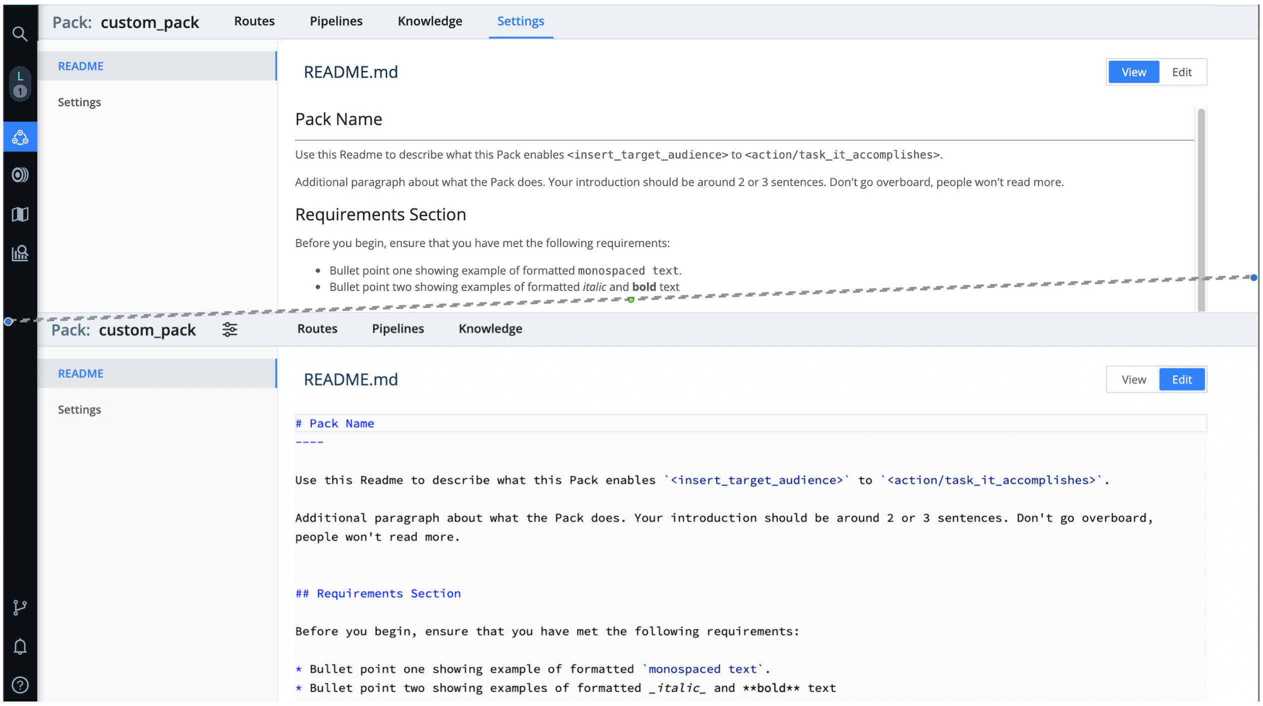
1. From the Pack's submenu, select Settings.



#	Route	Filter	Pipeline	
1	default	true	main	On

Pack Settings

2. To populate the Pack's README file, toggle **View** to **Edit**, replace the placeholder markdown content, and **Save**.



README.md

Pack Name

Use this Readme to describe what this Pack enables <insert_target_audience> to <action/task_it_accomplishes>.

Additional paragraph about what the Pack does. Your introduction should be around 2 or 3 sentences. Don't go overboard, people won't read more.

Requirements Section

Before you begin, ensure that you have met the following requirements:

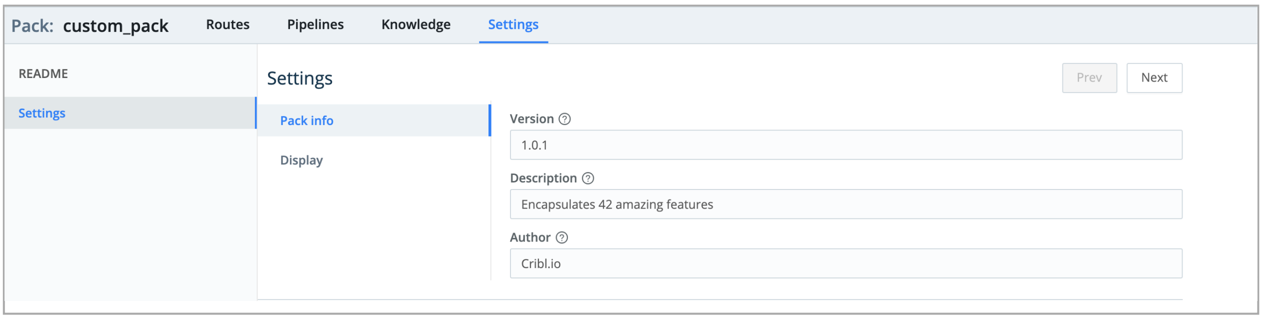
- Bullet point one showing example of formatted `monospaced text`.
- Bullet point two showing examples of formatted *italic* and **bold** text

README.md

```
# Pack Name
----

Use this Readme to describe what this Pack enables `<insert_target_audience>` to `<action/task_it_accomplishes>`.

Additional paragraph about what the Pack does. Your introduction should be around 2 or 3 sentences. Don't go overboard,
people won't read more.


## Requirements Section

Before you begin, ensure that you have met the following requirements:

* Bullet point one showing example of formatted `monospaced text`.
* Bullet point two showing examples of formatted _italic_ and **bold** text
```

Editing Pack's README

3. To update other metadata, click the left **Settings** tab.



Version: 1.0.1

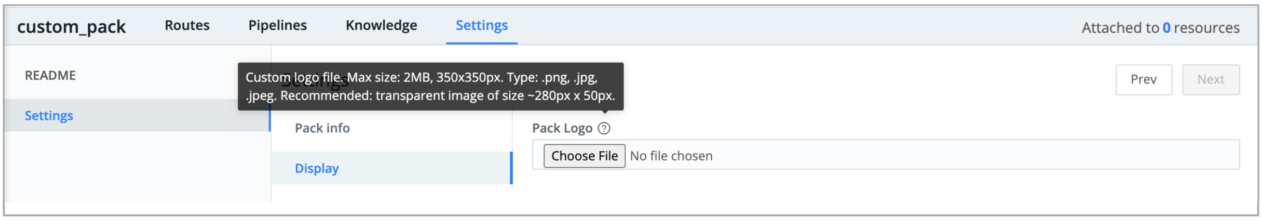
Description: Encapsulates 42 amazing features

Author: Cribl.io

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.



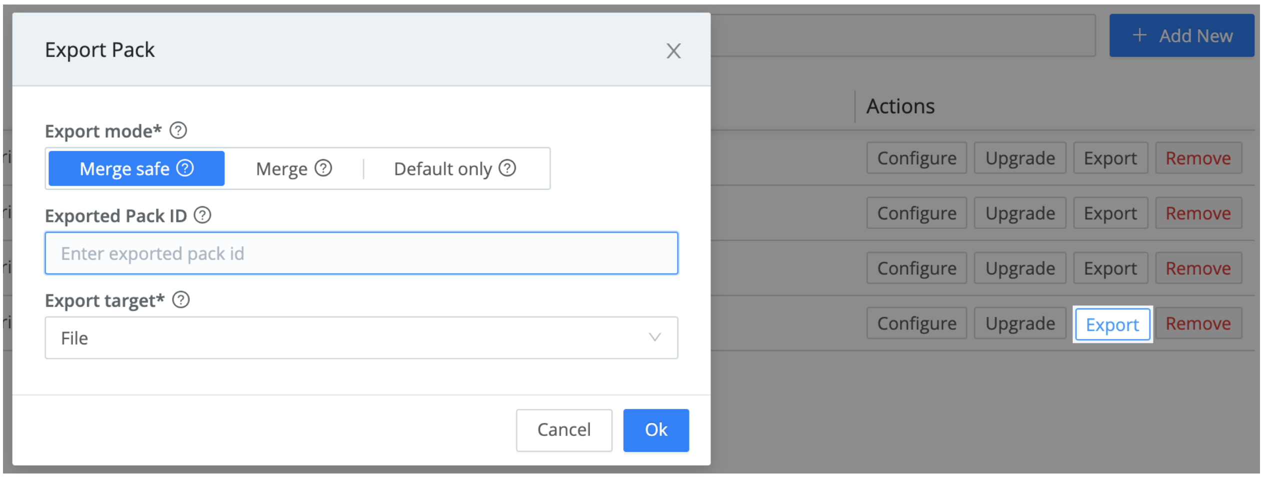
Custom logo file. Max size: 2MB, 350x350px. Type: .png, .jpg, .jpeg. Recommended: transparent image of size ~280px x 50px.

Pack Logo: No file chosen

Editing Pack's display (logo) settings

Exporting a Pack

To export a newly created or modified Pack, click its **Export** button on the Packs page.



Export Pack

Export mode*: Merge safe

Exported Pack ID: Enter exported pack id

Export target*: File

Exporting a Pack

The resulting Export Pack modal provides the following options.

Export Mode

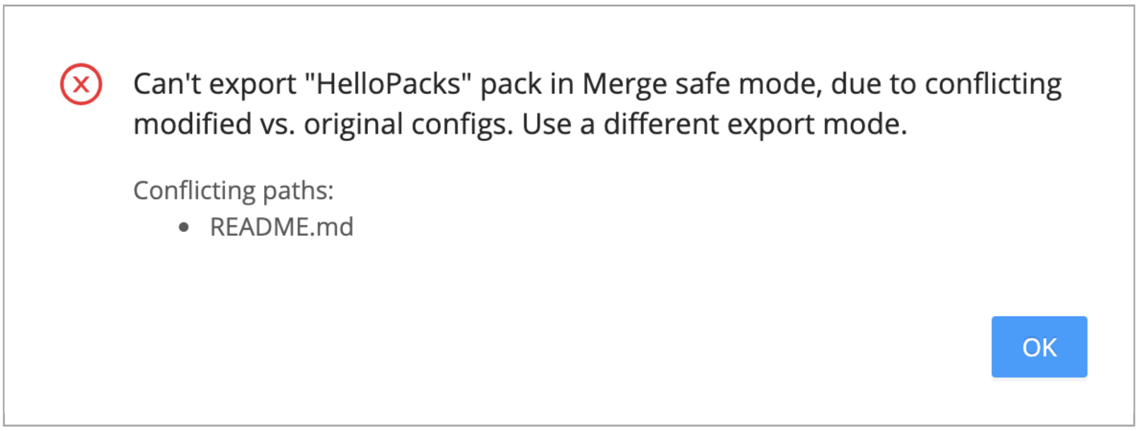
Select one of these three buttons:

- **Merge safe**: Attempt to safely merge local modifications into the Pack's default layer (original configuration), then export.
- **Merge**: Force-merge local modifications into the Pack's original configuration, then export.

  > **Merge** is the only export mode available when you've selected `Groups` as your Export target.

- **Default only**: Export only the Pack's original configuration, without local modifications.

The **Merge safe** option is conservative, and will block the export where Cribl 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.

Export Target

The options here are:

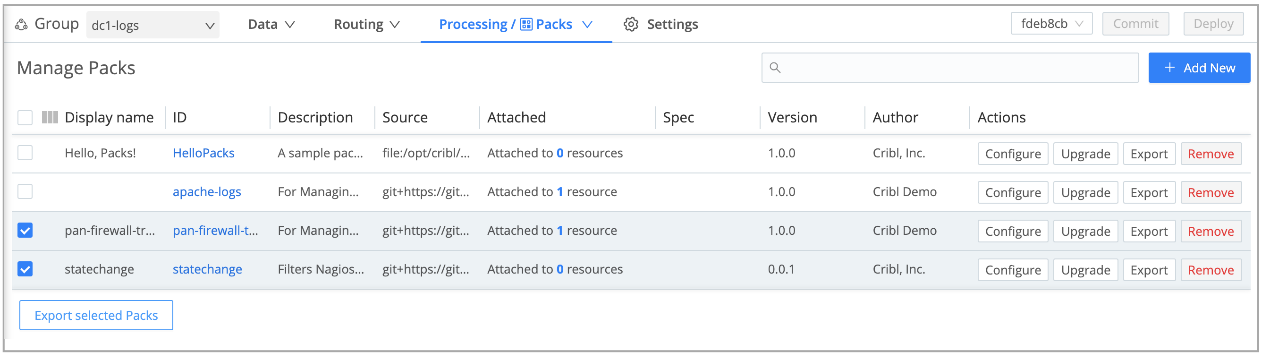
- **File** (the default): You'll be prompted to confirm a file name and destination after you click **OK**. (In Cribl Stream 3.5 and higher, the default file name automatically includes the Pack's version number.)

- **Groups**: Selecting this displays a **Groups** control, prompting you to select one or multiple existing Worker Groups/Fleets to export the Pack to. (The current Worker Group/Fleet is automatically omitted from the options.)

Exporting Multiple Packs

You can export multiple Packs in one operation, by selecting their check boxes and then clicking **Export multiple Packs**. This option comes with a few constraints:

- You can export multiple Packs only to Groups (not to Files).
- Therefore, this option is available only in distributed deployments.
- The only export mode available is **Merge**.
- The **Exported Pack ID** field is disabled, and hidden.



☐	Display name	ID	Description	Source	Attached	Spec	Version	Author	Actions
☐	Hello, Packs!	HelloPacks	A sample pac...	file:/opt/cribl/...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade Export Remove
☐		apache-logs	For Managin...	git+https://git...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
☑	pan-firewall-tr...	pan-firewall-t...	For Managin...	git+https://git...	Attached to 1 resource		1.0.0	Cribl Demo	Configure Upgrade Export Remove
☑	statechange	statechange	Filters Nagios...	git+https://git...	Attached to 0 resources		0.0.1	Cribl, Inc.	Configure Upgrade Export Remove

Exporting multiple packs at once

A status modal will list any Packs that failed to export.

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/Fleet and installing it into another. The two Worker Groups/Fleets do **not** need to have the same Leader Node.

> About the Following Examples
>
> - The API calls here include Worker Group/Fleet names as path parameters.
> - The `curl` commands assume that you have set the `$token` environment variable to match the value of a bearer token. Of course, this is just one option for authentication. See the Authentication topic for others; adapt the example commands to suit your chosen approach.

Export via API

Adapt and run this **Export pack** API call, using the export mode of your choice:

```
GET /api/v1/m/<worker_group_name>/packs/<pack_name>/export?mode=merge
```

Export Example

Let's export a Pack named `goat-herd` from the `default` Worker Group/Fleet, and use the `>` redirect to write the exported Pack to a file named `goat-herd.crbl`:

```
curl -X GET -H "Authorization: Bearer $token"
'https://logstream:9000/api/v1/m/default/packs/goat-herd/export?mode=merge' > goat-
herd.crbl
```

This request returns an octet-stream attachment which is downloaded as a `crbl` file. And voilà, you have exported your Pack.

Install via API

Installing the exported Pack in a different Worker Group/Fleet is a two-step process: First upload, then actually install.

Install via API – Step 1

Adapt and run this **Upload pack** API call, referencing the exported Pack file:

```
PUT /api/v1/m/<new_worker_group>/packs?filename=<pack_name>.crbl
```

Install Example – Step 1

We'll use our target Worker Group/Fleet name (in this example, it's `group420`). Then we need to specify the exported pack contents as a file payload, using the `--data-binary` option to upload the binary data without modification. The `@` prefix tells `curl` that `goat-herd.crbl` is the path to the file, not the data itself.

```
curl -X PUT -H "Authorization: Bearer $token"
'https://logstream:9000/api/v1/m/group420/packs?filename=goat-herd.crbl' --data-
binary "@goat-herd.crbl"
```

This request returns a JSON object of the following form:

```
{"source":"pack_name.random_id.crbl"}
```

Install via API – Step 2

Adapt and run this **Install pack** API call:

```
POST /api/v1/m/<new_worker_group>/packs
```

Meanwhile, remember that this API call will need a payload – the JSON object returned by the previous API call.

Install Example – Step 2

We'll use the `curl -d` option to specify the JSON object payload. We'll add a new element to the object, whose key is `id`, and whose value is the Pack's new name in the new Worker Group/Fleet.

Here, the `goat-herd` Pack is renamed as `billys_pack`. (If you do not wish to rename the Pack, just omit the `id` element – but keep the `source` element.)

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json"
'https://logstream:9000/api/v1/m/group420/packs' -d '{"source":"goat-
herd.987654321.crbl", "id":"billys_pack"}'
```

Copy via API

To bulk-copy Packs between Worker Groups/Fleets, adapt and run this **Clone pack** API call, referencing a source Worker Group/Fleet, destination Worker Group(s)/Fleet(s), and Pack(s).

```
POST /api/v1/packs/__clone__

{
  "srcGroup": "copy_from_this_worker_group_id",
  "dstGroups": [
    "destination_group_1",
    "destination_group_2",
    ...
  ],
  "packs": [
    "pack_id_1",
    "pack_id_2",
    ...
  ]
}
```

Copy Example

For example, to copy the Palo Alto Networks and Cisco ASA Packs from the default to `dc1-logs` and `dc2-logs` Worker Group/Fleets:

```
curl -X POST -H "Authorization: Bearer $token" -H "Content-Type: application/json"
'https://logstream:9000/api/v1/packs/__clone__' -d
'{"srcGroup":"default","dstGroups":["dc1-logs","dc2-logs"],"packs":
["PAN","cribl_cisco_asa_cleanup"]}'
```

Last updated by: Dritan Bitincka

;

5.5.1. 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 three-step process:

1. Prepare and Produce.

   In this initial phase, feel free to share with other Community members, who can help refine the Pack. For this development phase, consider working collaboratively on Cribl's Dispensary GitHub repo.

2. Publish the Pack to the Cribl Packs Dispensary™.

   The submission process, outlined below, validates that all required fields are included: Pack name, version, author, and license, if the version is newer than the last one published.

3. Celebrate!

> If you've already posted completed Packs to Cribl's GitHub repo, we encourage you to now submit them to the Packs Dispensary™. See Publishing a Pack.

Community Pack Guidelines

A Cribl Community Pack must be useful, reusable, and subject to the Cribl Pack Developer Agreement (PDA).

Making Your Pack Useful

Above all, what makes a Cribl Community Pack useful is the value that it provides for your fellow Community members. A Pack will be most useful if it includes Pipelines, along with supporting sample files and Knowledge objects (especially Lookups) as needed.

Making Your Pack Reusable

Reusability means that the Pack brings value to multiple Cribl Stream users. To make this possible, you should provide:

1. Instructions for using the Pack, including details on how to configure any relevant Sources and Destinations.

2. Details about the impact on downstream systems, so that users can prepare for changes that the Pack will make to data flowing through.

Acknowledging the Pack Developer Agreement

The Pack Developer Agreement (PDA) appears when you first access the Cribl Packs Dispensary™. As a Pack author, you must electronically acknowledge that you have read the Agreement, and that you intend to adhere to its requirements.

Before you Begin

Before you start creating your Pack, check the Cribl Packs Dispensary™ to see if something similar has already been published. If your idea for a Pack is new:

- Post to Cribl Community Slack's the `#packs` channel, asking whether any of your fellow Community members are working on a Pack that's similar to yours.

- If someone is already working along the same lines, then you have a good opportunity to collaborate on the Pack you want to create.

Read the docs that explain how to easily create your Pack from within the Cribl Stream UI.

Creating the Pack

Here's how you should formulate the information you'll need to create the Pack.

Pack names **should not**:

- Start with `Cribl` or `Cribl-`, which are reserved for Cribl-created packs.
- Use the word `Pack`, at all.

Pack names **should**:

- Start with `cc` (this indicates that the Pack was contributed by a Cribl Community member).

- Use all lowercase.
- Use dashes to separate words, e.g., `cc-tanium-events.`

Pack Version numbers should:

- Use `0.0.1` for the initial version of the Pack.
- Designate (number) subsequent versions as described in the Pack Publication Process below.

Pack Descriptions should:

- Be brief - no more than 1 or 2 sentences, e.g., `This Pack for Syslog inputs will reduce volume, and address timestamp normalization for Syslog senders that omit timezones.`

Pack Author names should be in the following format:

- Your Community name, a dash (`-`) and `cc`, e.g., `art chavez - cc.`

Pack License Notes should:

- Appear at the end of the Pack `README` and appropriately linked, e.g., `This Pack uses the following license: [Apache 2.0](https://github.com/criblio/appscope/blob/master/LICENSE).`

Creating Documentation for the Pack

To properly scope the documentation you write for your Pack, follow these principles:

- The optimal user experience is to have a single Pack that supports all data sets for the relevant device or sender.
- The Pack documentation should list all of the data sets that the Pack supports.
- If there are known data sets that are relevant but not yet supported, the documentation should list those, too.

For example, consider a device sending data to Cribl Stream, such as a Palo Alto Networks firewall. This device supports multiple data sets, in that Palo Alto Firewall data is really one co-mingled set of data that includes individual data sets like `PAN-Traffic`, `PAN-System`, `PAN-Accept`, and so on.

Think about what data sets would be involved for other devices, and how you would document them. For example, consider an F5 load balancer, various types of routers, or various types of servers. This exercise will help you anticipate what your users will need to see documented for your own Pack.

What a Pack Must Contain

Every Pack must contain some combination of Pipelines, samples, Routes, Knowledge objects (including Lookups), configuration descriptions, support contact information, and release notes. The exact ingredients will vary by Pack.

Pipelines

1. Except for rare cases, each Pack should have at least one Pipeline. Packs without Pipelines are of limited use.

2. Cribl recommends that you provide one Pipeline (and one Route) for each data set that your Pack supports.

3. Each Pipeline should have an internal **Comment** describing the overall functionality **and benefits** that the Pipeline provides. The procedure for adding a **Comment** is the same as for adding any other Function. **Comment** is under the **+ Function** > **Standard** drop-down.

4. Within each Pipeline, follow these best practices for Functions:

   - Use grouping to bundle any Functions that a user should enable or disable together.
   - Add a **Description** to each Function, to make it clear what is happening at each step, along with the Function's purpose and mechanics.

5. Overall, design for supportability and efficiency.

Samples

1. Include at least one data sample for each Pipeline. Data samples can be reused across Pipelines, but it is preferable to include multiple data samples, each specific to a Route/filter available in the Pipeline.

2. Remember to remove all sensitive data (e.g., internal host names or IP addresses) from samples.

Routes

1. A Pack includes Routes with appropriate filters.

2. Each Pipeline should have a corresponding Route on the Pack’s Routes page. (The only exception to this is when the Pack serves as a delivery mechanism for pre-processing and post-processing Pipelines.)

3. The filter in that Route should be as generic as possible. For example, if the data is coming from `PAN`, don’t assume you’ll have `sourcetype`. Instead, filter by `_raw.match()`.

4. Each Route must have a meaningful description.

5. The final Route (where **Filter** is set to `true`) should route to the `devnull` Pipeline.

Knowledge Objects

1. Include all Knowledge objects that the Pack requires, in the Pack –(Lookups, Parsers, Global Variables, Grok Patterns, and Schemas).

2. Remember to remove all sensitive data (e.g., internal host names, IP addresses) from Knowledge objects.

3. Lookups are especially important. If it's not practical to include the required Lookup, include instructions for building it.

Configuration Descriptions

Where applicable, the Pack documentation should include clear descriptions for each pre-shipped configuration.

Support Contact Information

The Pack documentation should list your preferred method of contact for support, e.g., your Cribl Community name for Slack DMs, or your email address.

Release Notes

If the Pack is an update from a previous release, the documentation should include Release Notes.

What a Pack Should Contain

A logo and a `README` file are recommended, but optional.

The Logo

You can include a logo associated with the technology addressed by the Pack, e.g., a Windows logo for Windows events, an AWS Cloudwatch logo for AWS Cloudwatch data, and so on.

The README

Including a `README` improves the user experience for your Pack. Create a `README.md` file in the **Settings** directory, with detailed answers to the following questions:

1. What does the Pack do, and why was it created? Here, you can state what value the Pack provides.

2. What technologies, data sources, and data destinations does the Pack interact with?

3. What other dependencies does the Pack have? As examples:

   - Does the Pack use an external tool, like Redis?
   - What is the minimum supported version of Cribl Stream? This must be newer than Cribl LogStream v.3.0.4 or Cribl Edge 3.3.0.
   - What deployment restrictions apply? What combination of single instance, distributed, or Cloud deployments of Cribl Stream does the Pack support?

4. What is required to configure a data source or destination?

   - Include specific examples of configurations when possible.
   - Link to specific sources and destinations to configure, e.g., link to AWS Firehose Source for a Pack that requires Firehose to collect data.

5. What else does your user need to know to use the Pack?

   - Include specific instructions.

6. How should your users contact you for support?

   - The `README` should contain the Pack author's contact info for providing feedback/requesting support, e.g., your Community name or email address.

Versioning Packs

Packs start at version `0.0.1`, and continue through as many "pre-release" versions as needed, until the authors feel that the Pack is ready for production use. During this initial **Prepare** phase, you'll share your idea with other Cribl Community members, to collaboratively refine the Pack.

Next, you'll enter the **Publish** phase, where it's appropriate to release version `1.0`. Here, you'll:

- Satisfy all the requirements for publishing a Pack, as documented above.

- Export the Pack by creating a `.crbl` file.
- Upload the Pack to the Cribl Packs Dispensary™, as outlined below. The Dispensary will validate the Pack name, version, author, and license, including by checking that the new version number is greater than the last one.

Versioning an Existing Pack

The easiest way to do this is to make the Pack changes in Cribl Stream, then export the new Pack and upload it to the Cribl Packs Dispensary™.

1. Update your Pack.

2. Increment the version number. See semantic versioning for guidelines.

3. In the `README`, update the **Release Notes** section to specify the new version and release date, and to describe what has changed. For a good model of how this is done, see the `README` on the Dispensary's **Microsoft Windows Events** Pack.

4. Export your Pack and save the `.crbl` file locally.

5. Upload your Pack's `.crbl` file. to the Cribl Packs Dispensary™, as outlined below. Remember to version the file.

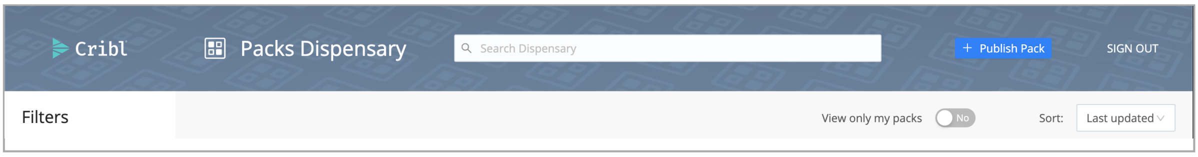
Publishing a Pack

To submit your Pack:

1. Sign into, or create an account on, the Cribl Packs Dispensary™ site. You can create an account using the same email address as used on your Cribl.Cloud account.

   If you don't have a Cribl.Cloud account, Cribl automatically creates a new one for you automatically when you create an account on the Cribl Packs Dispensary™.

2. Once signed in, you'll see the **+ Publish Pack** and the **View only my Packs** controls shown below.



Packs Dispensary™: Signed-in view

3. Click **+ Publish Pack**, and then click **Upload Pack**.

4. Select the `.crbl` Pack file you want to publish, and click **Submit**.

5. The Packs Dispensary™ will quickly verify whether the Pack has valid configurations and whether it meets all of the requirements outlined in this document.

6. If validation is successful, the Dispensary submits the Pack for review, and graduates it to a `Validating` state.

7. Once your Pack is reviewed, you'll receive an email from `packs@cribl.io` informing you whether it was accepted or rejected.

   Rejected Packs display on the Packs Dispensary™ as `Failed`, with a note about the rationale for rejection. After you've fixed your Pack, you can resubmit it.

Who Supports Packs

- Community-authored Packs are primarily supported by the Pack author, who also handles feature requests and suggestions.
- Cribl-authored Packs are supported by Cribl.
- For both kinds of Packs, the greater Cribl Slack Community also provides a wealth of knowledge - see the `#packs` channel.

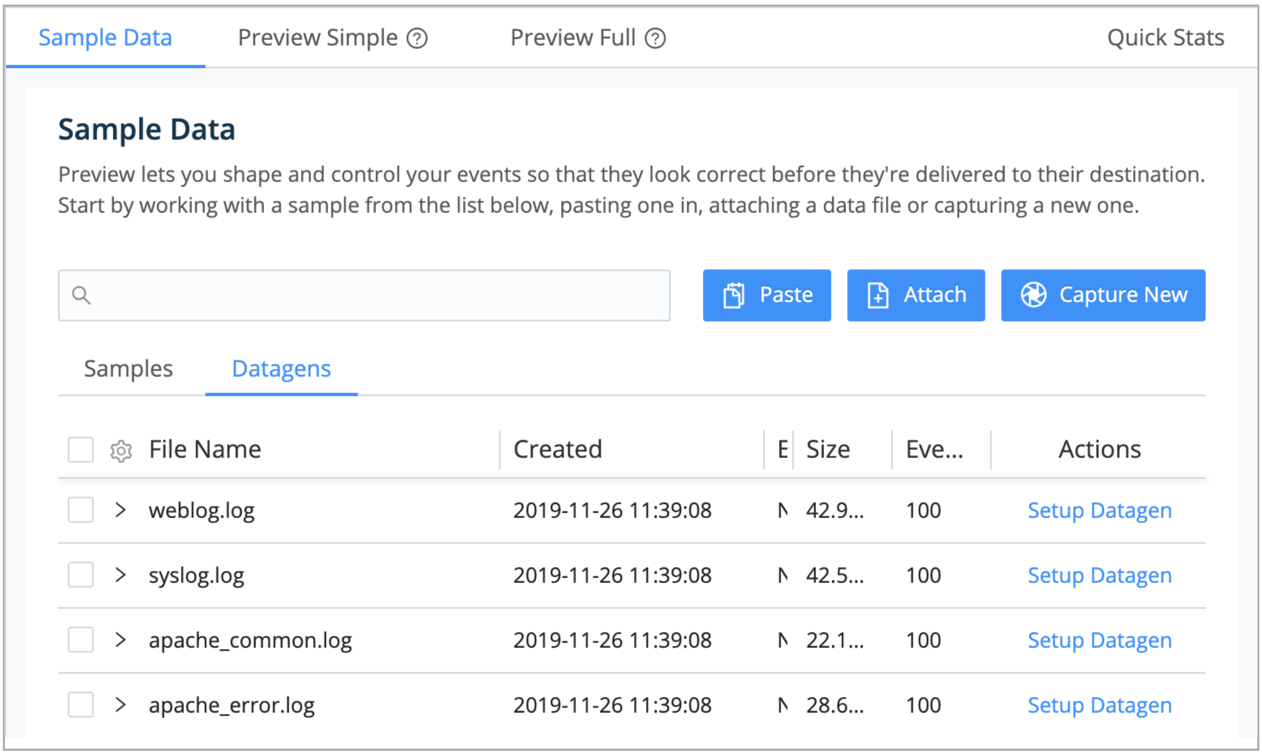
;

5.6. 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

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.

□	File Name	Created	E	Size	Eve...	Actions
□	> weblog.log	2019-11-26 11:39:08	N	42.9...	100	Setup Datagen
□	> syslog.log	2019-11-26 11:39:08	N	42.5...	100	Setup Datagen
□	> apache_common.log	2019-11-26 11:39:08	N	22.1...	100	Setup Datagen
□	> apache_error.log	2019-11-26 11:39:08	N	28.6...	100	Setup Datagen

Preview pane – add samples via paste, attach/upload file, or live capture

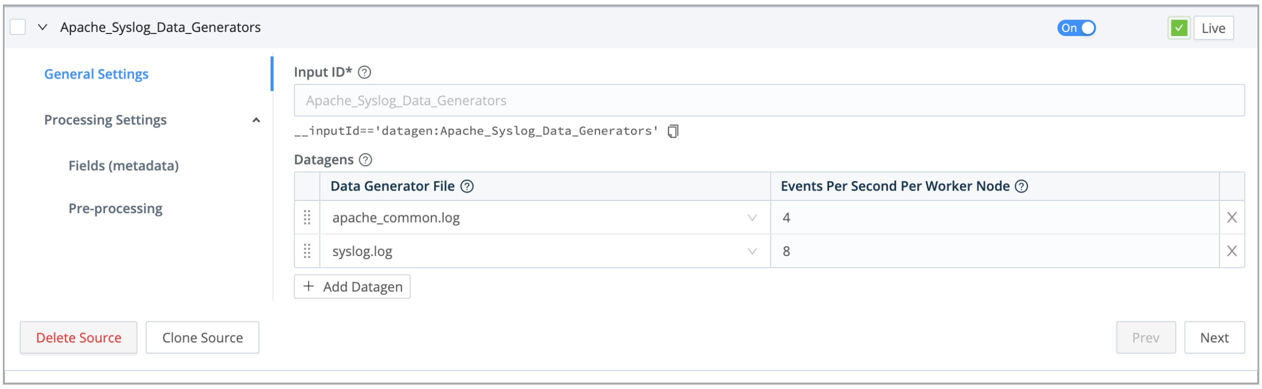
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen Source to use the template to generate real-time data at a given EPS (events per second) rate.

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 **+ Add New**.

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**.



Input ID*: Apache_Syslog_Data_Generators

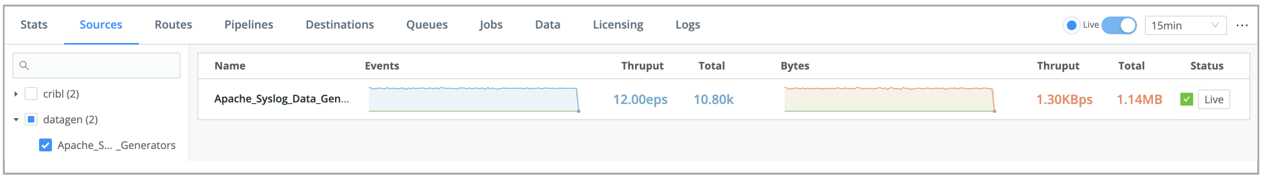
`__inputId=='datagen:Apache_Syslog_Data_Generators'`

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:

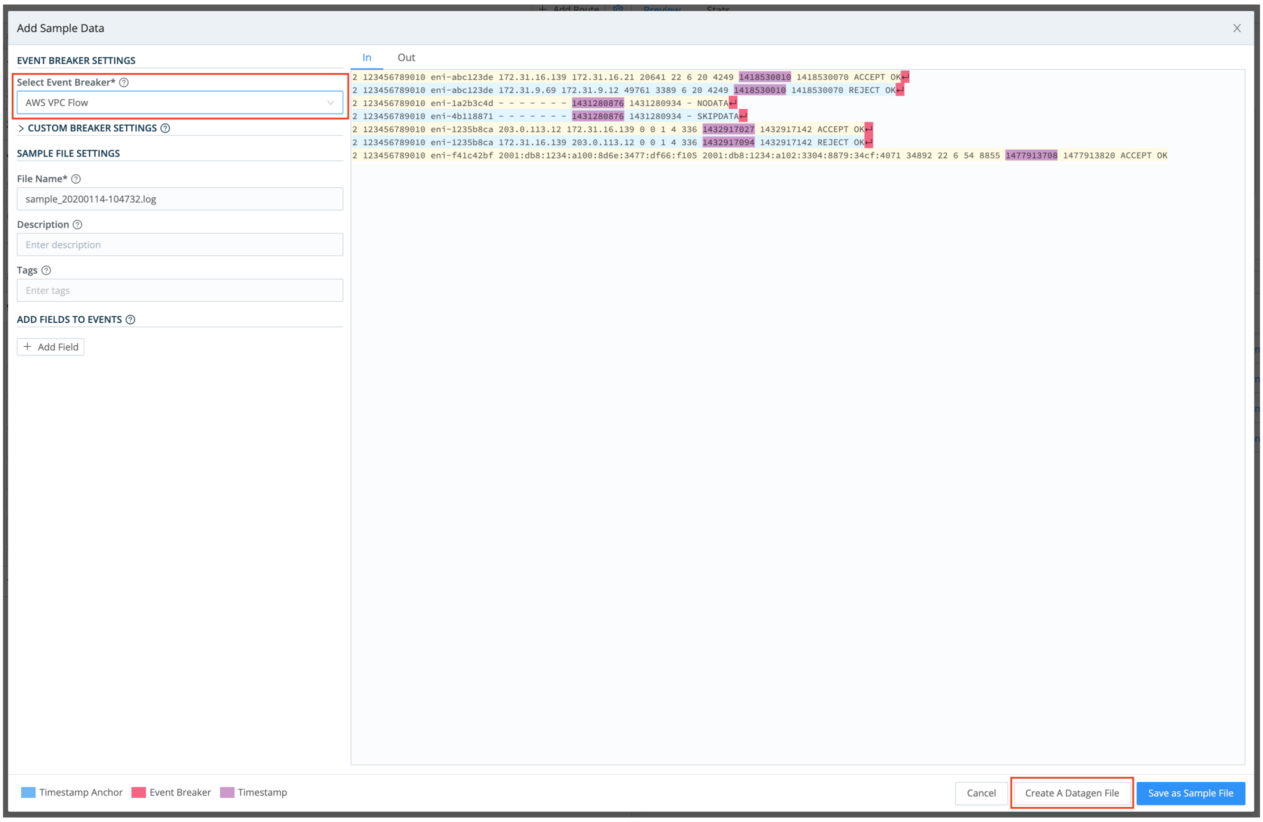
Go to **Preview** > **Paste a Sample**, and add a sample like the AWS VPC Flow logs below:

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010
1418530070 ACCEPT OK
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418530010
1418530070 REJECT OK
2 123456789010 eni-1a2b3c4d - - - - - - - 1431280876 1431280934 - NODATA
2 123456789010 eni-4b118871 - - - - - - - 1431280876 1431280934 - SKIPDATA
2 123456789010 eni-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027
1432917142 ACCEPT OK
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094
1432917142 REJECT OK
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105
2001:db8:1234:a102:3304:8879:34cf:4071 34892 22 6 54 8855 1477913708 1477913820
ACCEPT OK
```

From the **Event Breaker** drop-down, select **AWS VPC Flow** to ensure that:

- The pasted text gets broken properly into individual events (notice the Event Breaker on newlines).
- Timestamps are extracted correctly (text highlighted purple below).

Once you've verified these results, click **Create a Datagen File**.



Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: AWS VPC Flow

SAMPLE FILE SETTINGS

File Name*: sample_20200114-104732.log

Description: Enter description

Tags: Enter tags

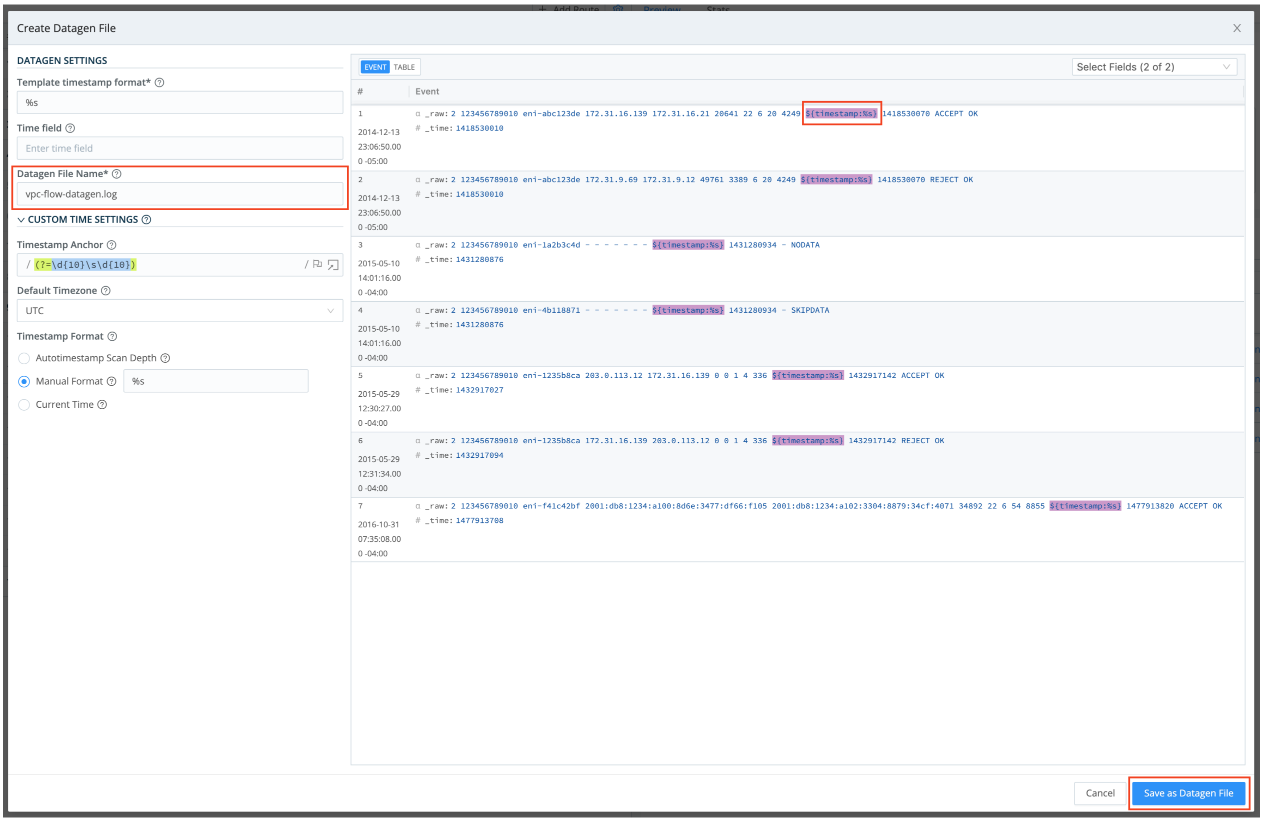
```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418530010 1418530070 ACCEPT OK
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418530010 1418530070 REJECT OK
2 123456789010 eni-1a2b3c4d - - - - - - - 1431280876 1431280934 - NODATA
2 123456789010 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

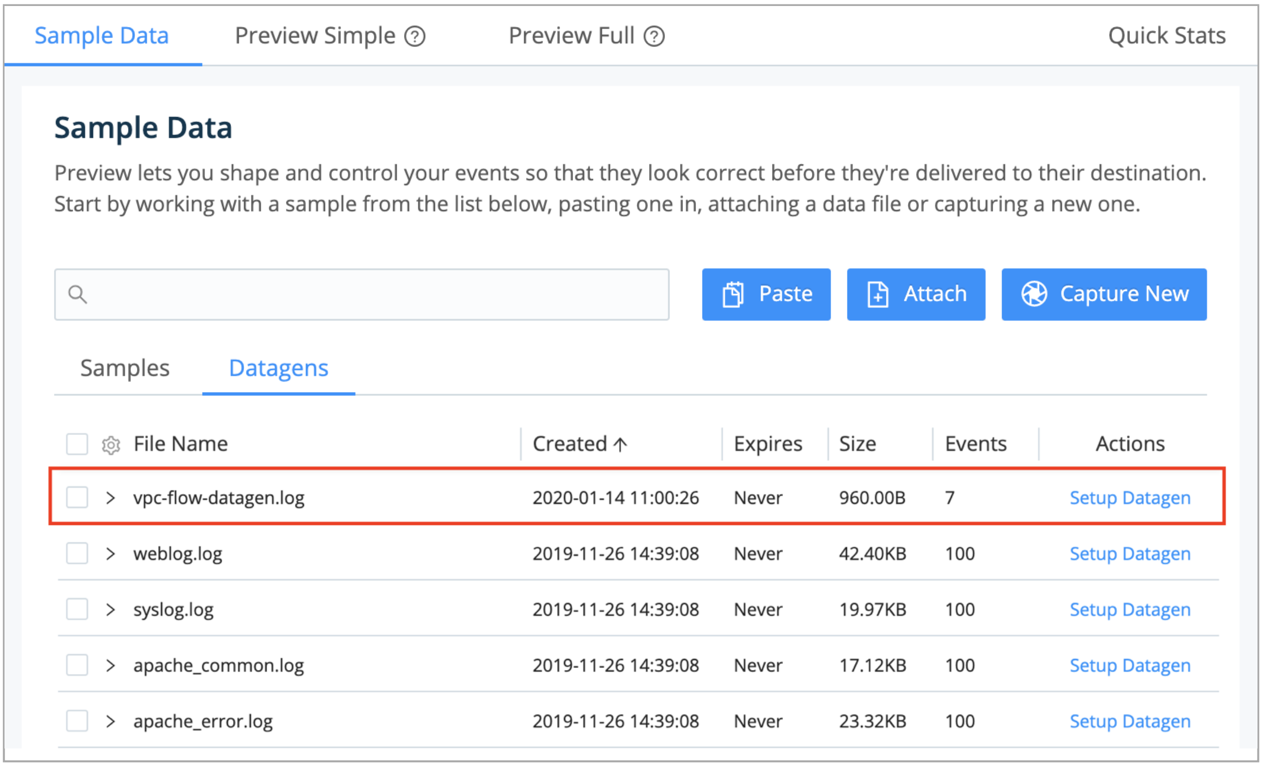
● Manual Format: %s

○ Current Time

```
#   Event
1   α _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
23:06:50.00
0 -05:00
2   α _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
23:06:50.00
0 -05:00
3   α _raw: 2 123456789010 eni-1a2b3c4d - - - - - - - ${timestamp:%s} 1431280934 - NODATA
    # _time: 1431280876
2015-05-10
14:01:16.00
0 -04:00
4   α _raw: 2 123456789010 eni-4b118871 - - - - - - - ${timestamp:%s} 1431280934 - SKIPDATA
    # _time: 1431280876
2015-05-10
14:01:16.00
0 -04:00
5   α _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
12:30:27.00
0 -04:00
6   α _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
12:31:34.00
0 -04:00
7   α _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
07:35:08.00
0 -04:00
```

Saving a named datagen template

To confirm that the datagen file has been created, check **Preview** > **Datagens**.



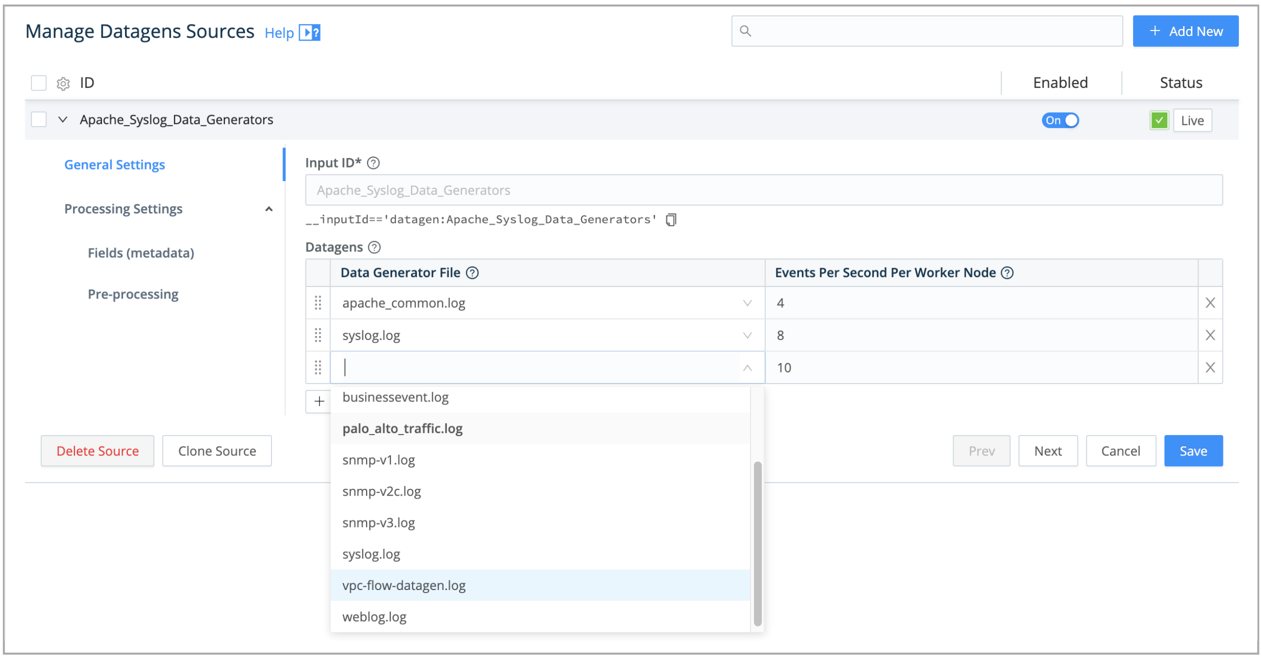
Sample Data

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.

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.



Input ID*

Apache_Syslog_Data_Generators

```
__inputId=='datagen:Apache_Syslog_Data_Generators'
```

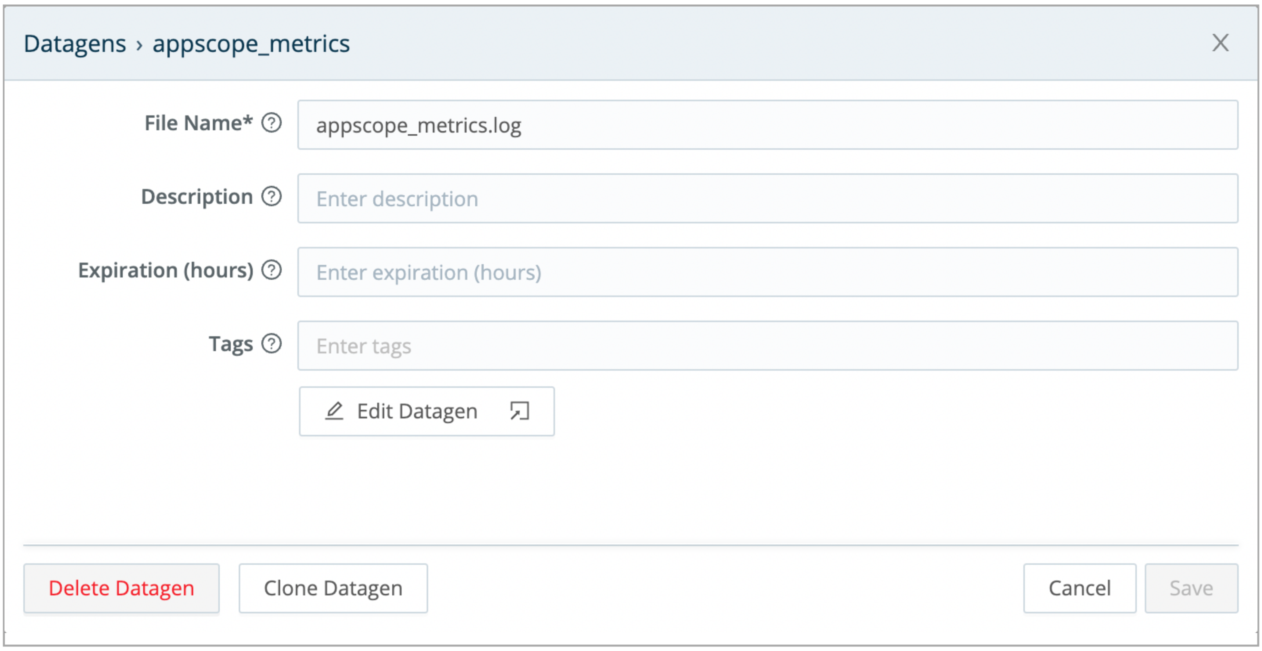
Datagens

Data Generator File	Events Per Second Per Worker Node
apache_common.log	4
syslog.log	8
	10

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

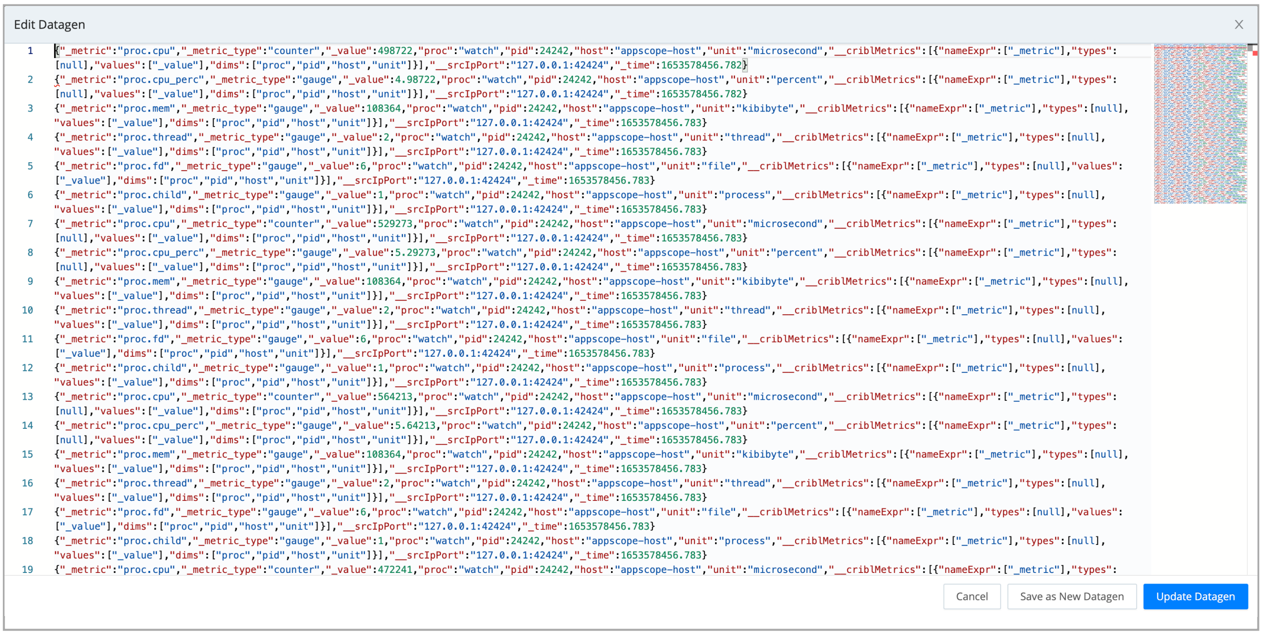
Description: Enter description

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

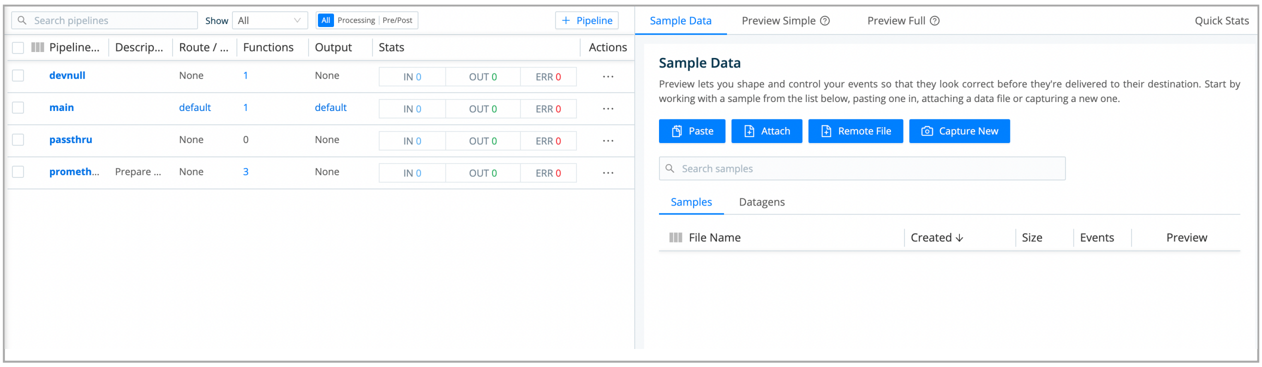
;

5.7. 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.



Pipeline...	Descrip...	Route / ...	Functions	Output	Stats			Actions
devnull		None	1	None	IN 0	OUT 0	ERR 0	···
main		default	1	default	IN 0	OUT 0	ERR 0	···
passthru		None	0	None	IN 0	OUT 0	ERR 0	···
prometh...	Prepare ...	None	3	None	IN 0	OUT 0	ERR 0	···

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.

File Name	Created ↓	Size	Events	Preview

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.)

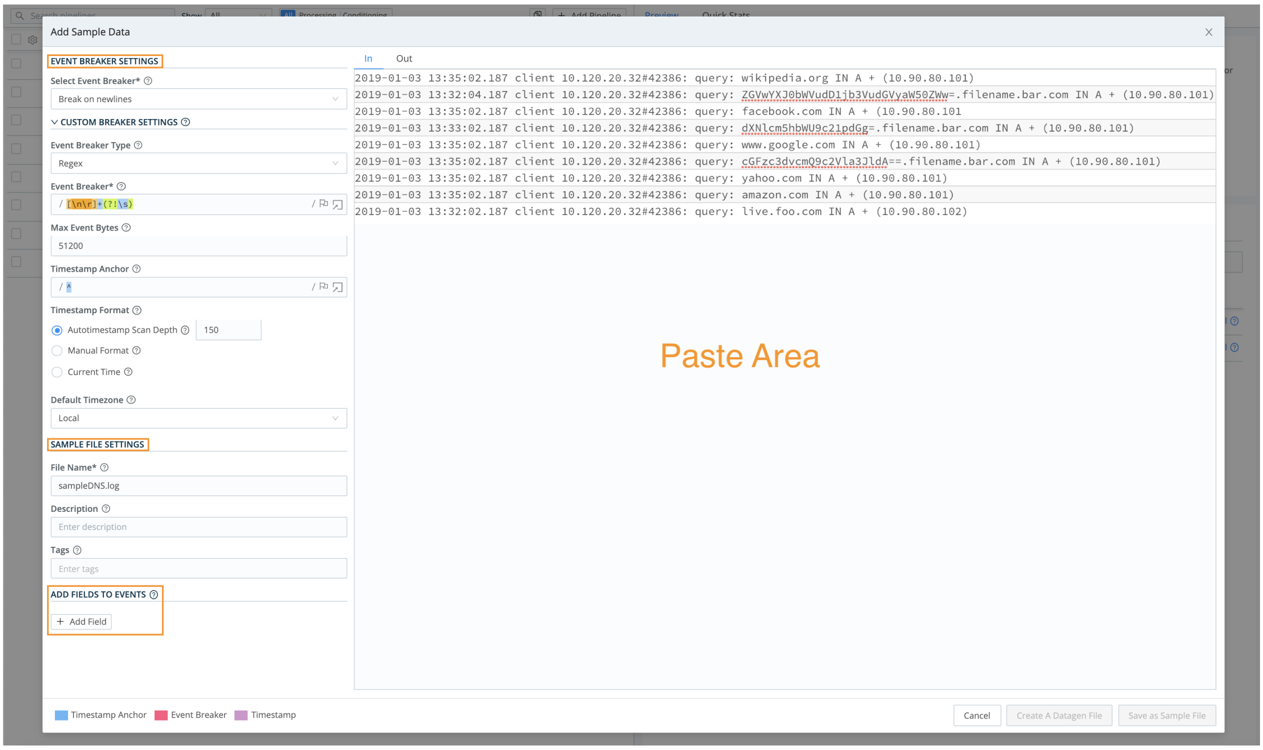
Adding Sample Data

When you're on the Pipelines or Routes page, you can add samples through any of the supported options: **Paste**, **Attach**, **Remote File**, or **Capture New**. The **Paste**, **Attach**, and **Remote File** options work with content that needs to be broken into events, while the **Capture New** option works with events only.

> The **Remote File** option requires a working Edge node, and is not available when you've teleported to the node.

Paste Area

Once you've clicked the **Paste** button, attached a file, or uploaded a remote file, you'll see an **Add Sample Data** modal, where you can edit and then save your data.



Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: Break on newlines

CUSTOM BREAKER SETTINGS

Event Breaker Type: Regex

Event Breaker*: /[\n\r]+(?!\s)/

Max Event Bytes: 51200

Timestamp Anchor: /^/

Timestamp Format

● Autotimestamp Scan Depth: 150

○ Manual Format

○ Current Time

Default Timezone: Local

SAMPLE FILE SETTINGS

File Name*: sampleDNS.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

```
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: wikipedia.org IN A + (10.90.80.101)
2019-01-03 13:32:04.187 client 10.120.20.32#42386: query: ZGVwYXJ0bWVudD1jb3VudGVyaW50ZWw=.filename.bar.com IN A + (10.90.80.101)
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: facebook.com IN A + (10.90.80.101
2019-01-03 13:33:02.187 client 10.120.20.32#42386: query: dXNlcm5hbWU9c21pdGg=.filename.bar.com IN A + (10.90.80.101)
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: www.google.com IN A + (10.90.80.101)
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: cGFzc3dvcmQ9c2Vla3JldA==.filename.bar.com IN A + (10.90.80.101)
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: yahoo.com IN A + (10.90.80.101)
2019-01-03 13:35:02.187 client 10.120.20.32#42386: query: amazon.com IN A + (10.90.80.101)
2019-01-03 13:32:02.187 client 10.120.20.32#42386: query: live.foo.com IN A + (10.90.80.102)
```

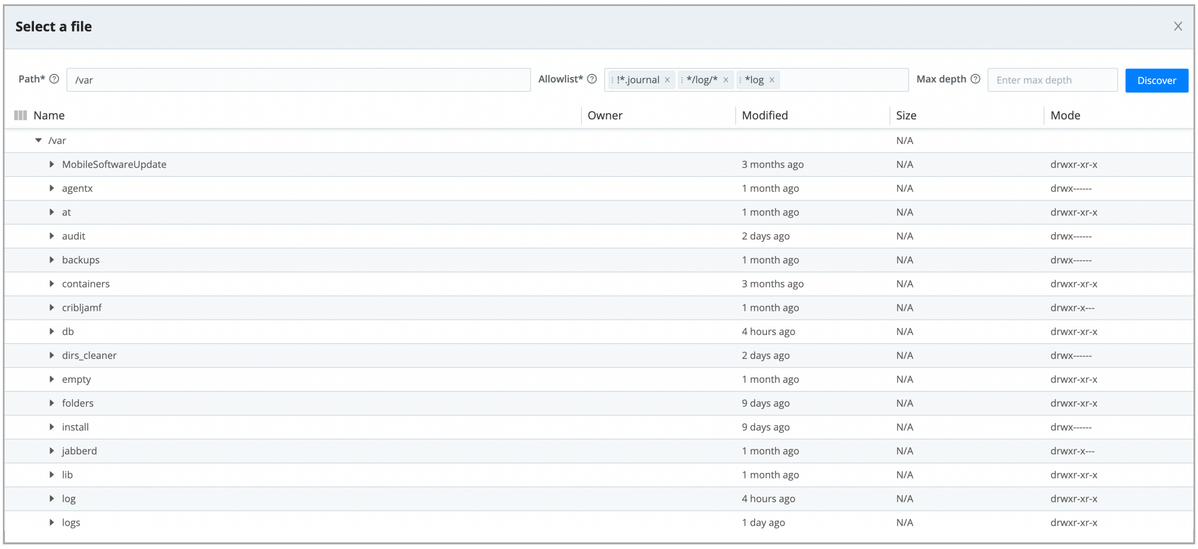
Add Sample Data modal

Remote File

To upload data from a file on an Edge node:

1. Click the **Remote File** 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

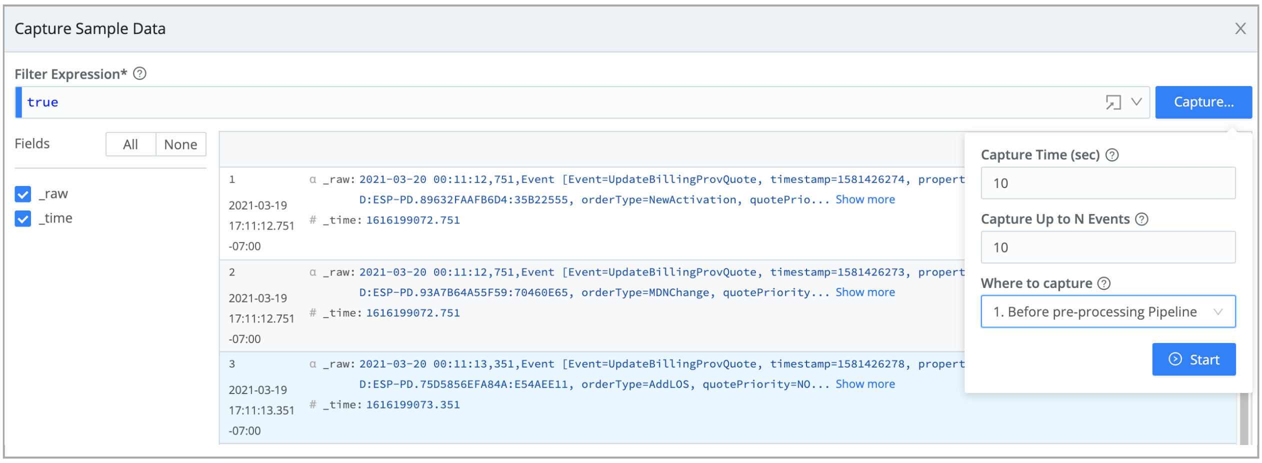
An Event Breaker is a regular expression that tells Cribl Stream how to break the file or pasted content into events. Breaking will occur at the **start** of the match. Cribl Stream ships with several common breaker patterns out of the box, but you can also configure custom breakers. The UI here is interactive, 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 New** button opens a slightly different modal – it does not require event breaking. In the composite screenshot below, we've already captured some events using the **Capture** drop-down.


Capture Sample Data

Filter Expression*: `true`

Fields: ☑ _raw ☑ _time

```
1    α _raw: 2021-03-20 00:11:12,751,Event [Event=UpdateBillingProvQuote, timestamp=1581426274, propert
             D:ESP-PD.89632FAAFB6D4:35B22555, orderType=NewActivation, quotePrio... Show more
2021-03-19
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
             D:ESP-PD.93A7B64A55F59:70460E65, orderType=MDNChange, quotePriority... Show more
2021-03-19
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
             D:ESP-PD.75D5856EFA84A:E54AEE11, orderType=AddLOS, quotePriority=NO... Show more
2021-03-19
17:11:13.351 # _time: 1616199073.351
-07:00
```

Capture Time (sec): 10

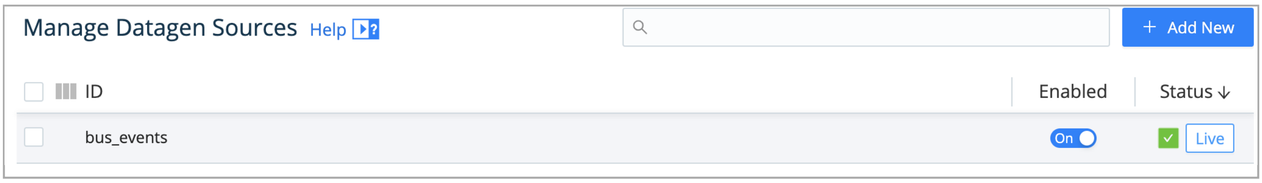
Capture Up to N Events: 10

Where to capture: 1. Before pre-processing Pipeline

Capture New > Capture Sample Data modal

Capturing from a Single Source or Destination

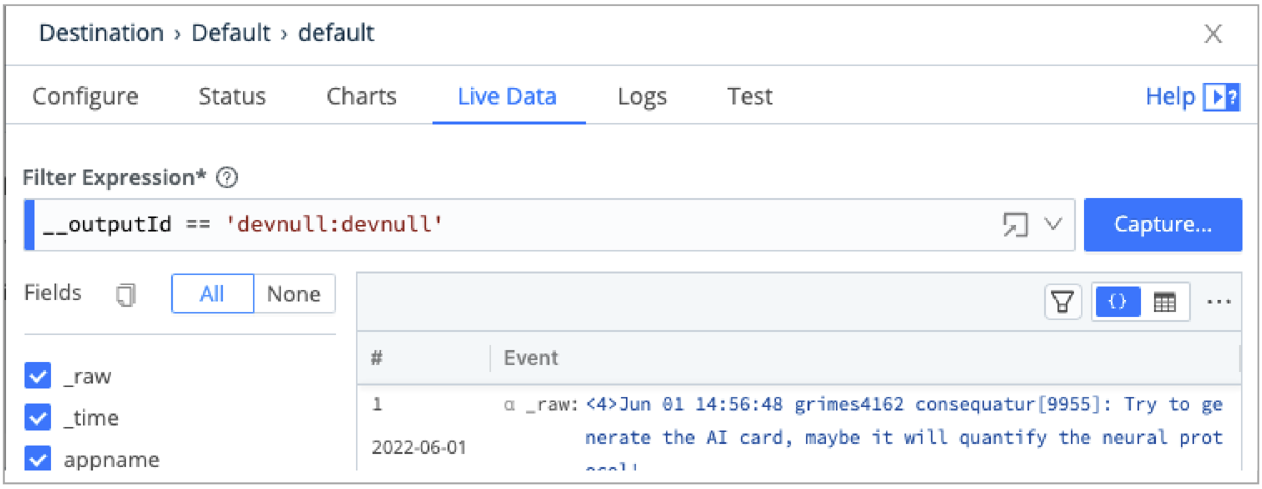
To capture data from a single enabled Source or Destination, it's fastest to use the Sources or Destinations UI instead of the Preview pane. You can initiate an immediate capture by clicking the **Live** button on the Source's or Destination's configuration row.


Manage Datagen Sources

□ ID	Enabled	Status ↓
□ bus_events	On	✓ Live

Source > Live button

You can similarly start an immediate capture from within an enabled Source's or Destination's configuration modal, by clicking the modal's **Live Data** tab.


Destination › Default › default

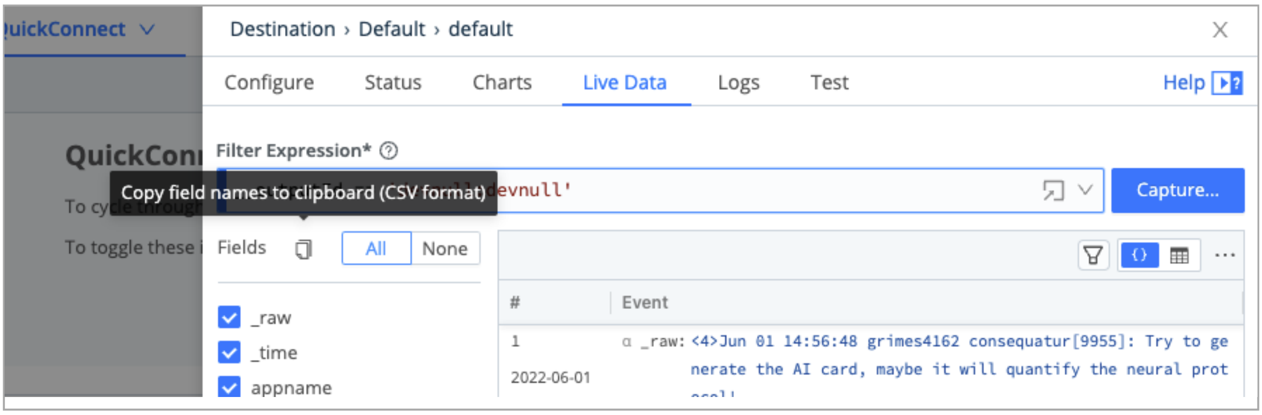
Filter Expression*: `__outputId == 'devnull:devnull'`

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
                     ocol!
```

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           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 - Copy Fields Icon

Controlling Sample Size

To prevent in-memory samples from getting unreasonably large, samples input by any means (Paste, Attach, Remote File, or Capture New) are constrained by a limit set at global ⚙ **Settings** (lower left) > **General Settings** > **Limits** > **Max sample size**. The default limit is `256KB`, and you can adjust this upward 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 Sample 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:

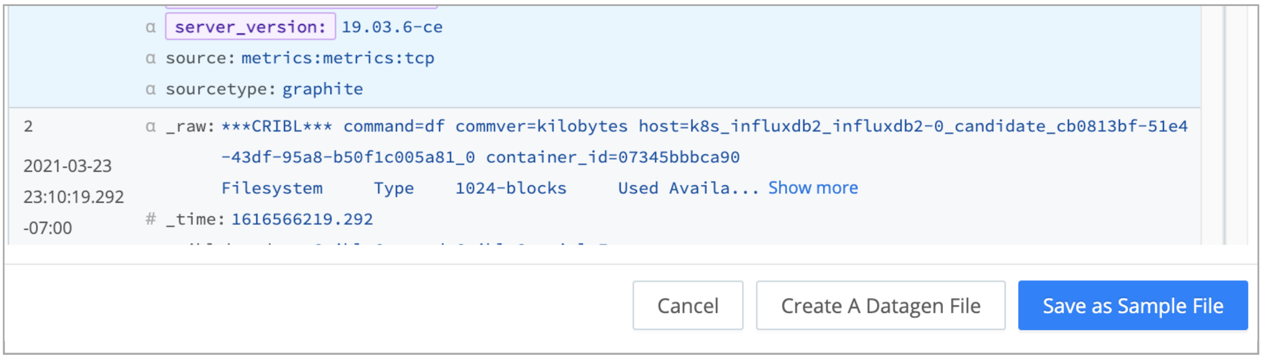
SYMBOL	MEANING
α	string
#	numeric
b	boolean
m	metric
{}	JSON object
[]	array

On JSON objects and arrays, you'll also see:

SYMBOL	MEANING
+	expandable
-	collapsible

Saving Sample Data

The Preview pane's **Add Sample Data** or **Capture Sample 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.

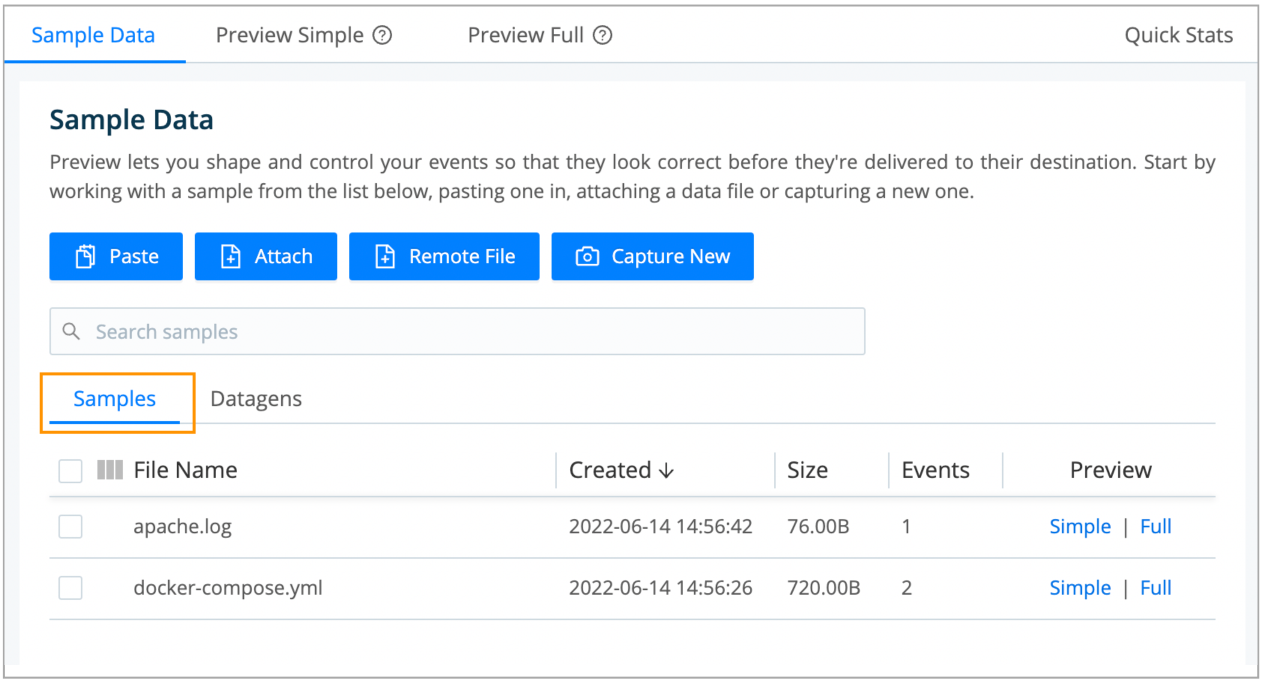


```
              α server_version: 19.03.6-ce
              α source: metrics:metrics:tcp
              α sourcetype: graphite
2             α _raw: ***CRIBL*** command=df commver=kilobytes host=k8s_influxdb2_influxdb2-0_candidate_cb0813bf-51e4
2021-03-23          -43df-95a8-b50f1c005a81_0 container_id=07345bbbca90
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

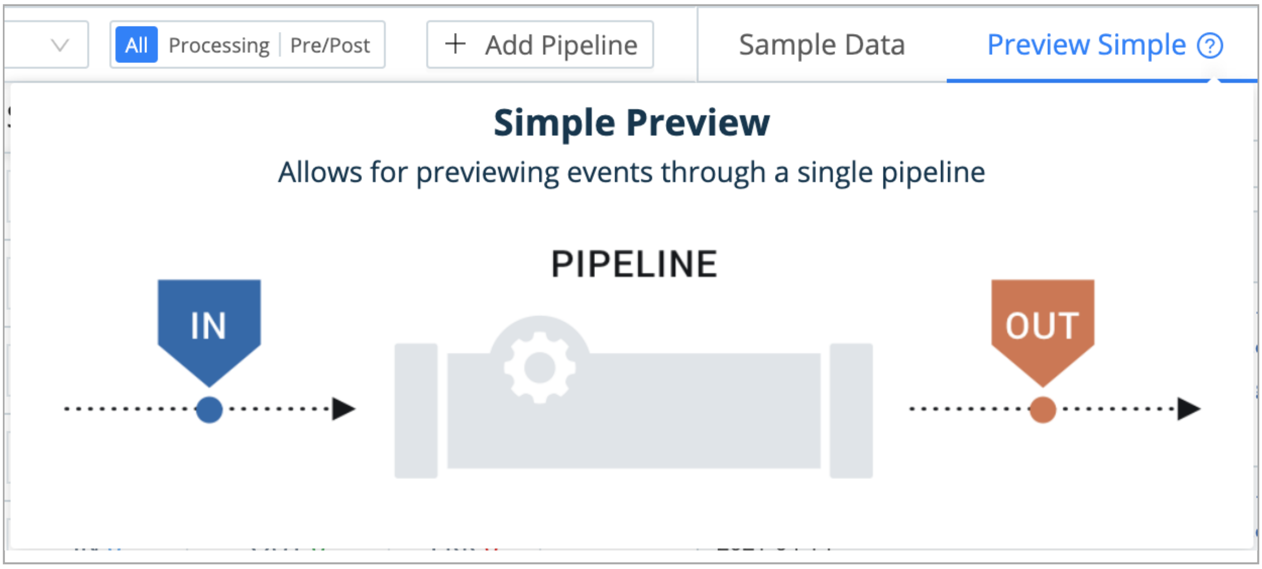
Preview lets you shape and control your events so that they look correct before they're delivered to their destination. Start by working with a sample from the list below, pasting one in, attaching a data file or capturing a new one.

□	File Name	Created ↓	Size	Events	Preview
□	apache.log	2022-06-14 14:56:42	76.00B	1	Simple | Full
□	docker-compose.yml	2022-06-14 14:56:26	720.00B	2	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 **Preview Simple** option enables you to view events on either the **IN** or the **OUT** (processed) side of a single Pipeline.

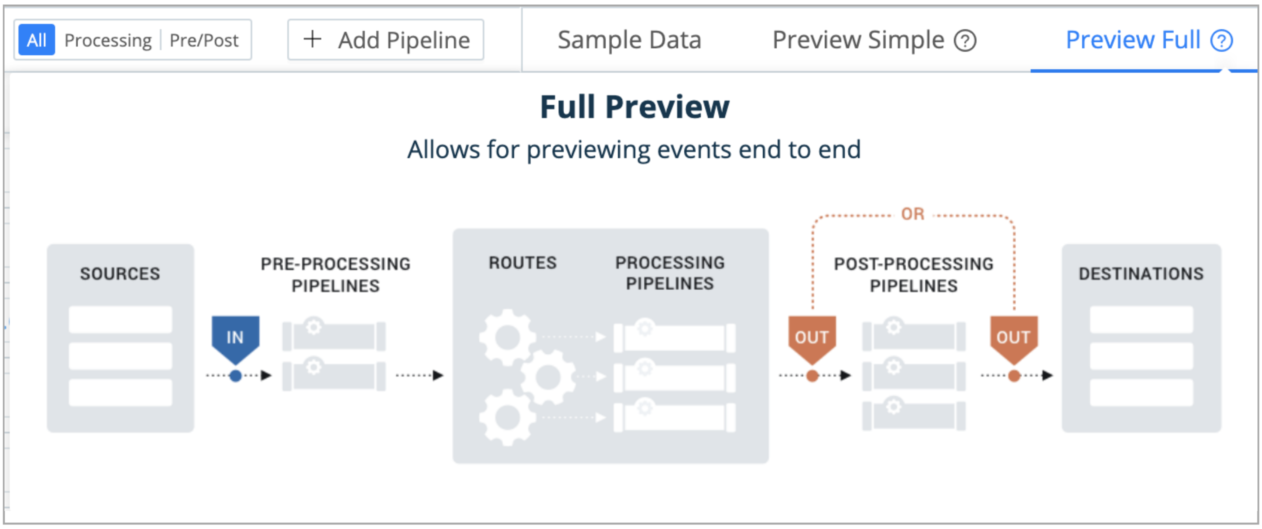


Simple Preview

Allows for previewing events through a single pipeline

Preview Simple schematic

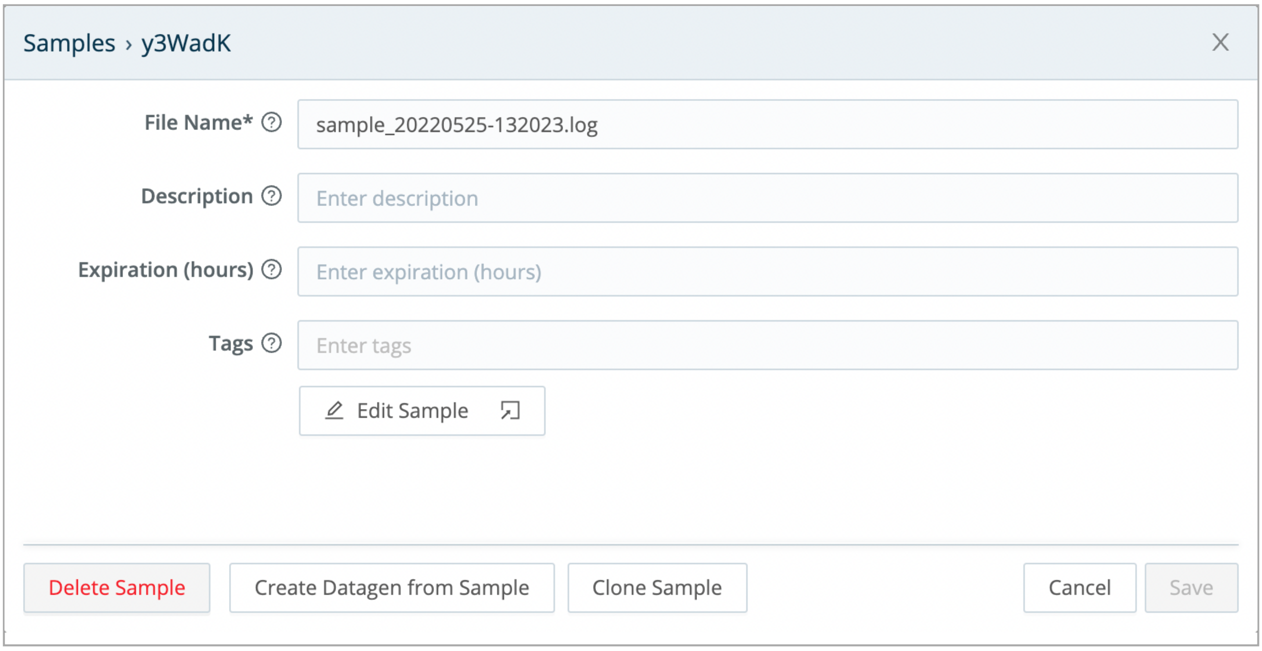
The **Preview Full** option gives you a choice of viewing events on the **OUT** side of either the processing or post-processing Pipeline. Selecting this option expands the Preview pane's upper controls to include an **Exit Point** drop-down, where you make this choice.



Preview Full 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

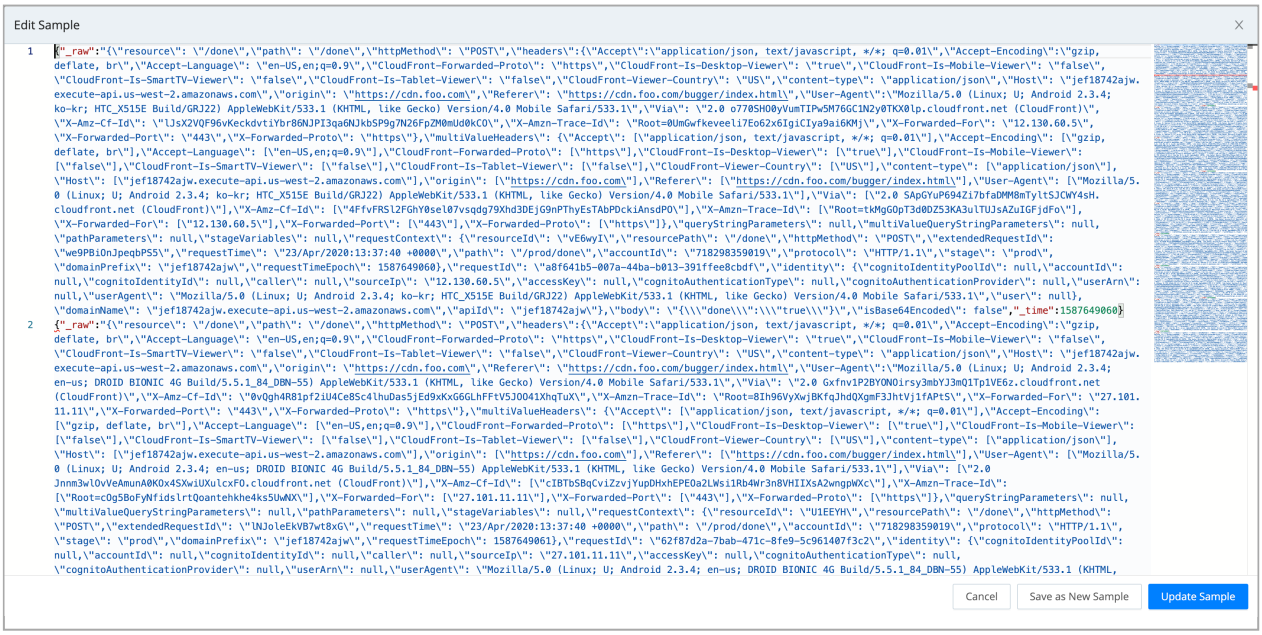
Description ⓘ Enter description

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.

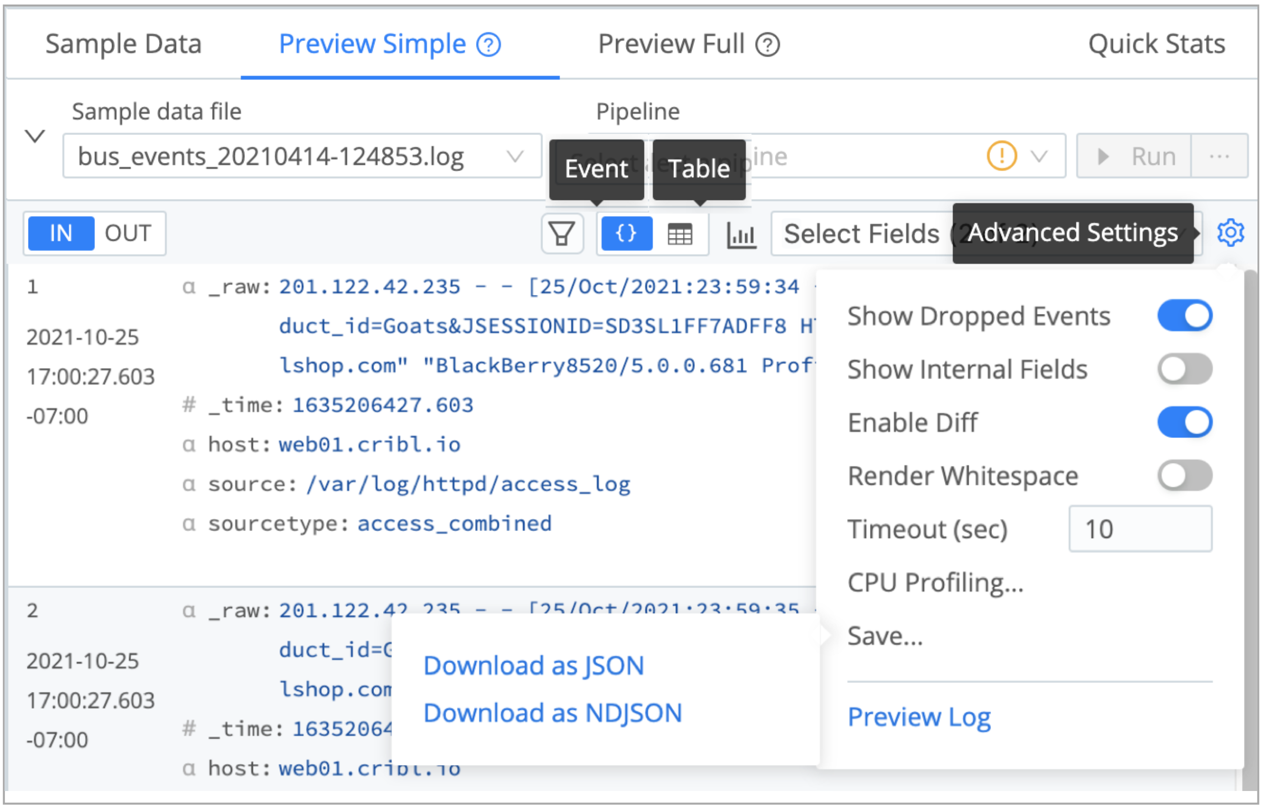


```
{"_raw":"{\"resource\": \"/done\",\"path\": \"/done\",\"httpMethod\": \"POST\",\"headers\":{\"Accept\":\"application/json, text/javascript, */*; q=0.01\",\"Accept-Encoding\":\"gzip,
deflate, br\",\"Accept-Language\": \"en-US,en;q=0.9\",\"CloudFront-Forwarded-Proto\": \"https\",\"CloudFront-Is-Desktop-Viewer\": \"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:



```
1             α _raw: 201.122.42.235 - - [25/Oct/2021:23:59:34
                      duct_id=Goats&JSESSIONID=SD3SL1FF7ADFF8 H
2021-10-25            lshop.com" "BlackBerry8520/5.0.0.681 Prof
17:00:27.603  # _time: 1635206427.603
-07:00        α host: web01.cribl.io
              α source: /var/log/httpd/access_log
              α sourcetype: access_combined

2             α _raw: 201.122.42.235 - - [25/Oct/2021:23:59:35
                      duct_id=C
2021-10-25            lshop.com
17:00:27.603  # _time: 16352064
-07:00        α host: web01.cribl.io
```

Show Dropped Events ●
Show Internal Fields ○
Enable Diff ●
Render Whitespace ○
Timeout (sec) 10

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 · .

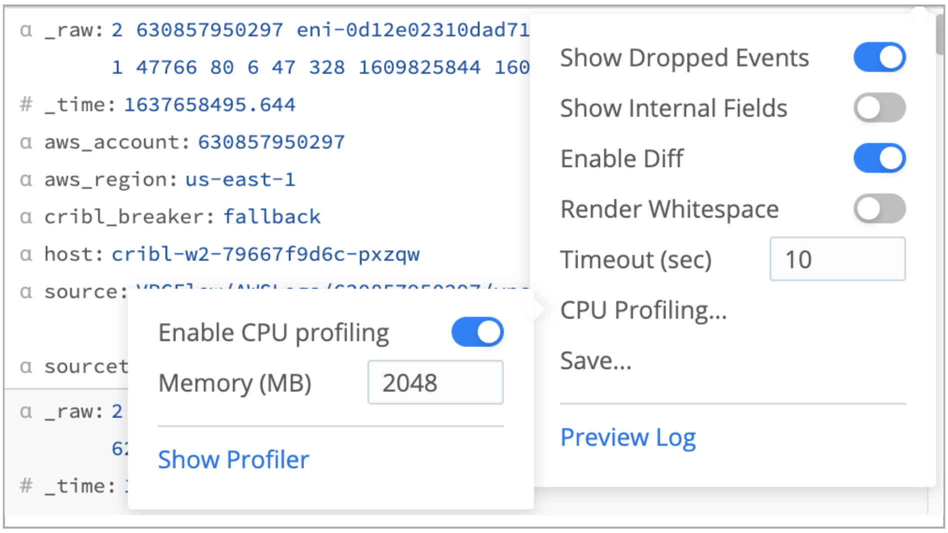
Timeout (Sec)

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.

CPU Profiling

The **CPU Profiling** submenu offers an **Enable CPU Profiling** slider, which in turn unlocks a **Memory (MB)** limit control. If very large data samples fail to load, you can enable the Profiler and adjust the defaults.

For example, you might increase the **Timeout** limit (described above) to `30`, and the **Memory (MB)** limit to `3048`. Optionally, click **Show Profiler** to see detailed results in a modal.



```
α _raw: 2 630857950297 eni-0d12e02310dad71
        1 47766 80 6 47 328 1609825844 160
# _time: 1637658495.644
α aws_account: 630857950297
α aws_region: us-east-1
α cribl_breaker: fallback
α host: cribl-w2-79667f9d6c-pxzqw
α source:
α sourcet
α _raw: 2
        6
# _time:
```

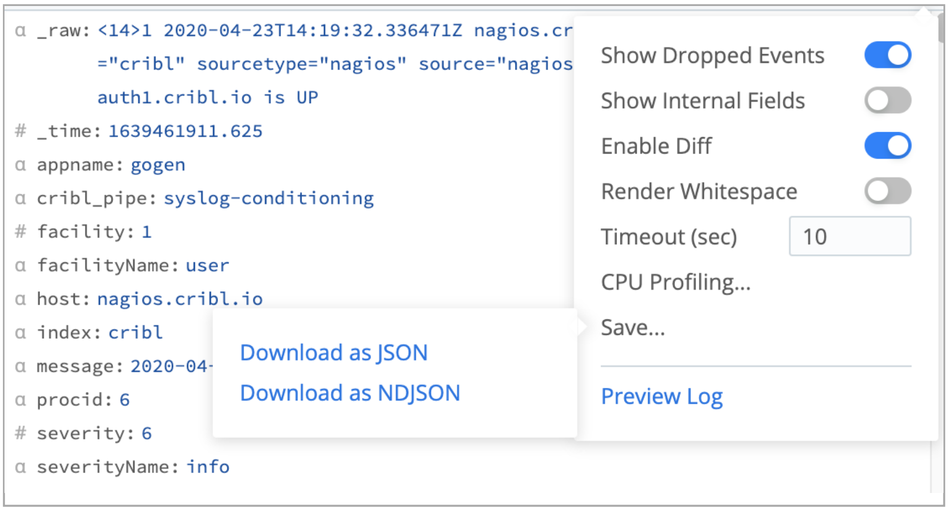
Show Dropped Events: on
Show Internal Fields: off
Enable Diff: on
Render Whitespace: off
Timeout (sec): 10

Enable CPU profiling: on
Memory (MB): 2048

CPU 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
```

Show Dropped Events: on
Show Internal Fields: off
Enable Diff: on
Render Whitespace: off
Timeout (sec): 10

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

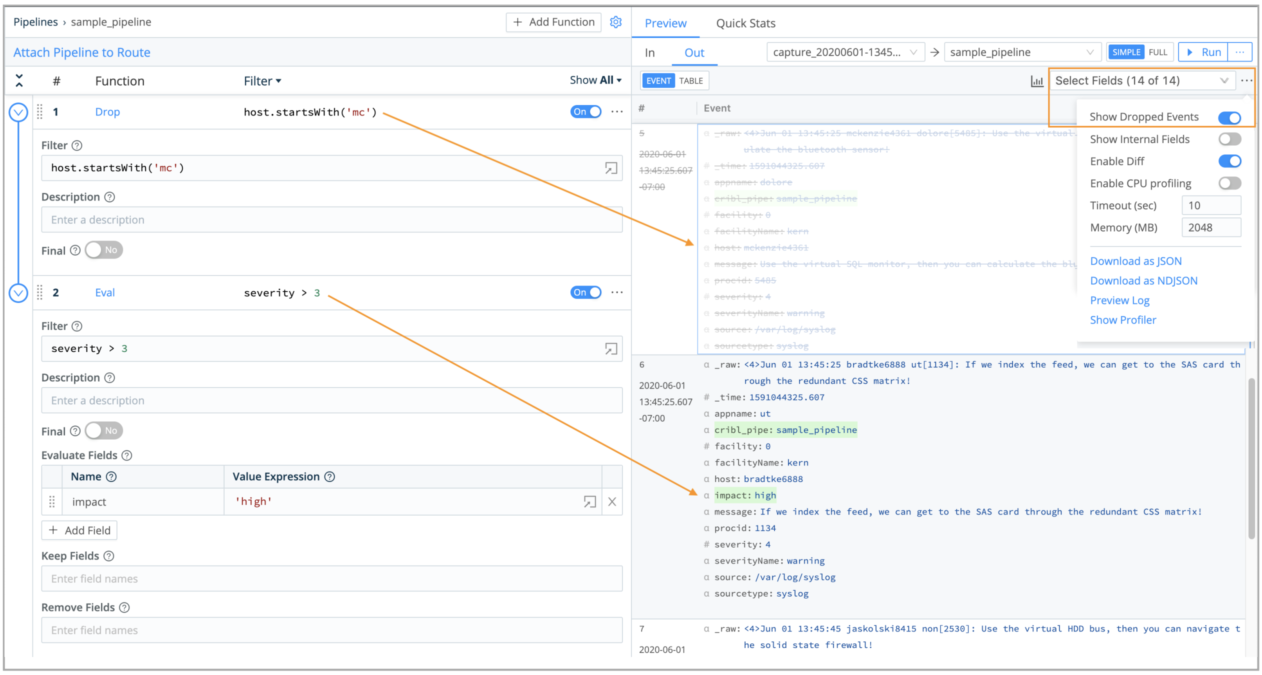
As data traverses through a Pipeline's Functions, events can be modified, and some might be dropped altogether. The **OUT** tab indicates changes using this color coding:

- **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** slider.

- **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.



#	Function	Filter	
1	Drop	host.startsWith('mc')	On
2	Eval	severity > 3	On

Function 1 (Drop):
Filter: host.startsWith('mc')
Description: Enter a description
Final: No

Function 2 (Eval):
Filter: severity > 3
Description: Enter a description
Final: No
Evaluate Fields:

Name	Value Expression
impact	'high'

Keep Fields: Enter field names
Remove Fields: Enter field names

capture_20200601-1345... → sample_pipeline

Select Fields (14 of 14)

Show Dropped Events: On
Show Internal Fields: Off
Enable Diff: On
Enable CPU profiling: Off
Timeout (sec): 10
Memory (MB): 2048

```
#  Event
~~5~~
~~2020-06-01 13:45:25.607 -07:00~~
   ~~_raw: <4>Jun 01 13:45:25 mckenzie4361 dolore[5485]: Use the virtual~~
         ~~ulate the bluetooth sensor!~~
   ~~_time: 1591044325.607~~
   ~~appname: dolore~~
   ~~cribl_pipe: sample_pipeline~~
   ~~facility: 0~~
   ~~facilityName: kern~~
   ~~host: mckenzie4361~~
   ~~message: Use the virtual SQL monitor, then you can calculate the blu~~
   ~~procid: 5485~~
   ~~severity: 4~~
   ~~severityName: warning~~
   ~~source: /var/log/syslog~~
   ~~sourcetype: syslog~~
6
2020-06-01 13:45:25.607 -07:00
   _raw: <4>Jun 01 13:45:25 bradtke6888 ut[1134]: If we index the feed, we can get to the SAS card th
         rough the redundant CSS matrix!
   _time: 1591044325.607
   appname: ut
   cribl_pipe: sample_pipeline
   facility: 0
   facilityName: kern
   host: bradtke6888
   impact: high
   message: If we index the feed, we can get to the SAS card through the redundant CSS matrix!
   procid: 1134
   severity: 4
   severityName: warning
   source: /var/log/syslog
   sourcetype: syslog
7
2020-06-01
   _raw: <4>Jun 01 13:45:45 jaskolski8415 non[2530]: Use the virtual HDD bus, then you can navigate t
         he solid state firewall!
```

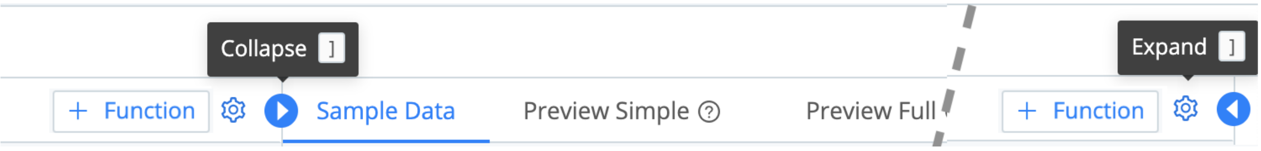
Dropped and added fields in a Pipeline's output

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

With the Routes or Pipelines page displayed in the left pane, hover over the pane divider (in the headers row) to display the Collapse/Expand toggle shown in the composite screenshot below.



Collapse / Expand toggle (composite)

Click Collapse to hide the Preview pane. This allows the Route or Pipeline configuration to expand to your browser's full width. (The Preview pane collapses automatically on narrow viewports.)

Click Expand at your browser's right edge to restore the split view. The pane divider will snap back to wherever you last dragged it.

;

5.8. 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

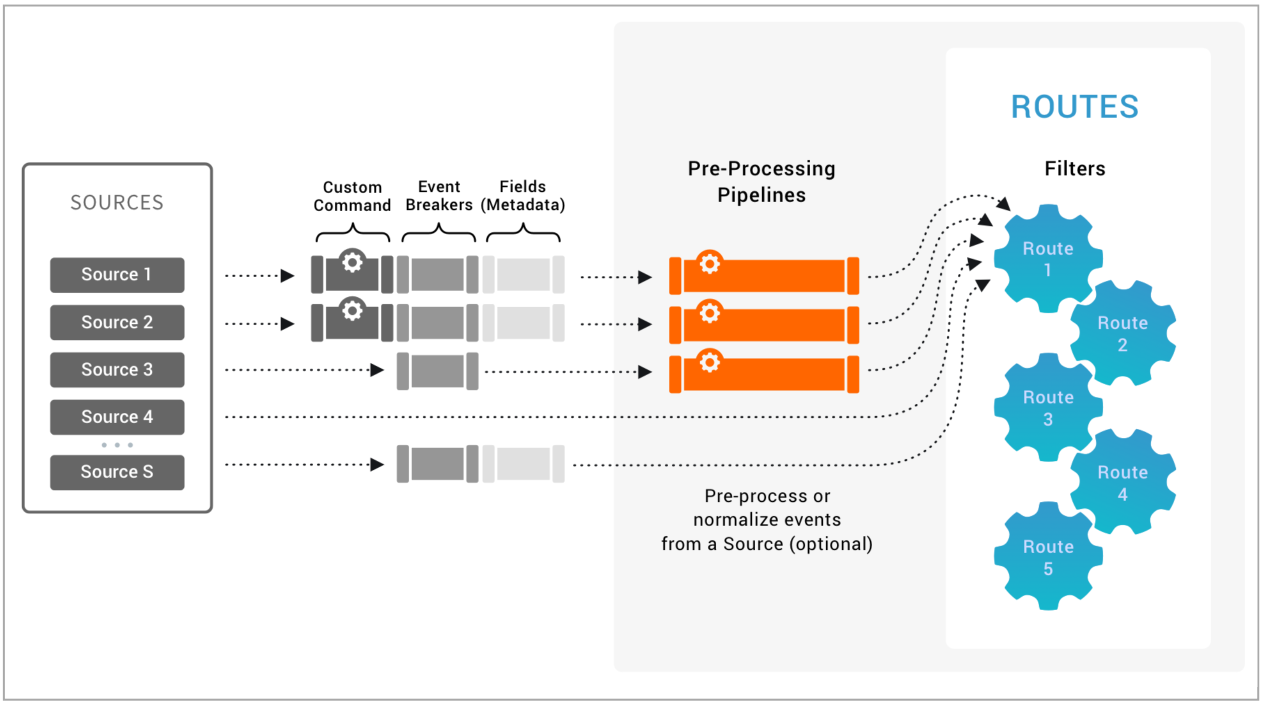
Use a sample of your real data in Data Preview. Sample data can come from a sample Source file that you upload or paste into Cribl Stream.

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.

;

6. Functions

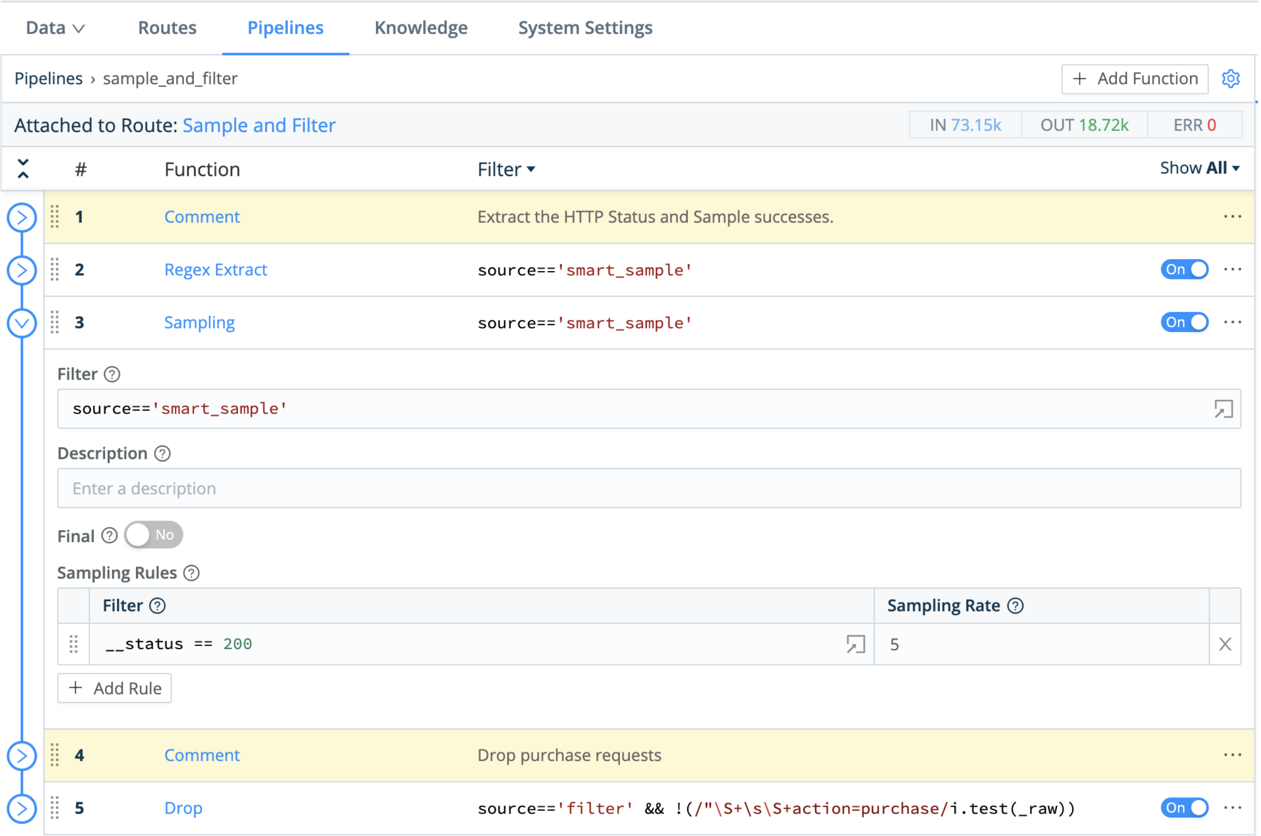
When events enter a Pipeline, they're processed by a series of Functions. At its core, a Function is code that executes on an event, and it encapsulates the smallest amount of processing that can happen to that event.

The term "processing" means a variety of possible options: string replacement, obfuscation, encryption, event-to-metrics conversions, etc. For example, a Pipeline can be composed of several Functions – one that replaces the term `foo` with `bar`, another one that hashes `bar`, and a final one that adds a field (say, `dc=jfk-42`) to any event that matches `source=='us-nyc-application.log'`.

How Do They Work

Functions are atomic pieces of JavaScript code that are invoked on each event that passes through them. To help improve performance, Functions can be configured with filters to further scope their invocation to matching events only.

You can add as many Functions in a Pipeline as necessary, though the more you have, the longer it will take each event to pass through. Also, you can turn Functions **On**/**Off** within a Pipeline as necessary. This enables you to preserve structure as you optimize or debug.



Attached to Route: Sample and Filter

IN 73.15k | OUT 18.72k | ERR 0

#	Function	Filter	
1	Comment	Extract the HTTP Status and Sample successes.	
2	Regex Extract	`source=='smart_sample'`	On
3	Sampling	`source=='smart_sample'`	On

Filter: `source=='smart_sample'`

Description: Enter a description

Final: No

Sampling Rules

Filter	Sampling Rate
`__status == 200`	5

#	Function	Filter	
4	Comment	Drop purchase requests	
5	Drop	`source=='filter' && !(/"\S+\s\S+action=purchase/i.test(_raw))`	On

Functions stack in a Pipeline

You can reposition Functions up or down the Pipeline stack to adjust their execution order. Use a Function's left grab handle to drag and drop it into place.

The `Final` Toggle

Similar to the `Final` toggle in Routes, the `Final` toggle here controls the flow of events at the Function level. Its states are:

- `No` (**default**): means that matching events processed by this Function will be passed down to the next Function.
- `Yes`: means that this Function is the last one that will be applied to matching events. All Functions further down the Pipeline will be skipped. A Function with `Final` set to `Yes` will display an **F** indicator in the Pipeline stack.

Functions and Shared-Nothing Architecture

Cribl 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.

This is particularly important to understand for certain Functions that might imply state-sharing, such as Aggregations, Sampling, Dynamic Sampling, Suppress, etc.

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 `@`.

Custom Functions

For an overview of adding custom Functions to Cribl Stream, see our blog post, Extending Cribl: Building Custom Functions.

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

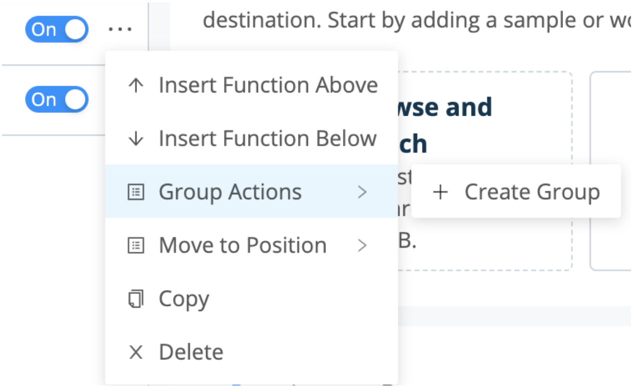
- Aggregate events in real-time (i.e., statistical aggregations): Aggregations
- Convert events to metrics format: Publish Metrics, Prometheus Publisher (beta)
- Resolve hostname from IP address: Reverse DNS (beta)
- Extract numeric values from event fields, converting them to type `number`: Numerify
- 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

Function Groups

A Function group is a collection of consecutive Functions that can be moved up and down a Pipeline's Functions stack together. Groups help you manage long stacks of Functions by streamlining their display. They are a UI visualization only: While Functions are in a group, those Functions maintain their global position order in the Pipeline.

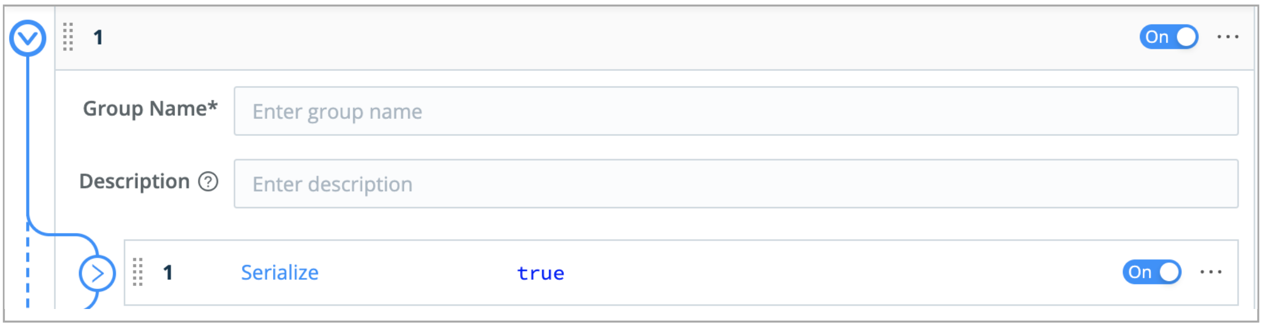
> Function groups work much like Route groups.

To build a group from any Function, click the Function's ••• (Options) menu, then select **Group Actions > Create Group**.



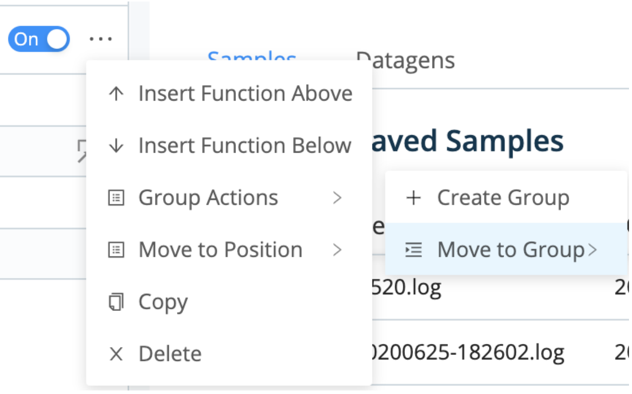
Creating a group

You'll need to enter a **Group Name** before you can save or resave the Pipeline. Optionally, enter a **Description**.



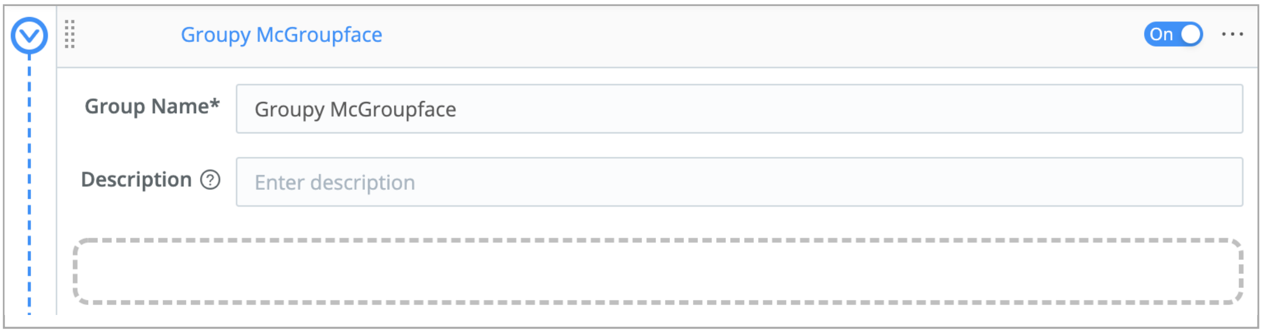
Naming a group

Once you've saved at least one group to a Pipeline, other Functions' ••• (Options) > **Group Actions** submenus will add options to **Move to Group** or **Ungroup/Ungroup All**.



Expanded Group Actions submenu

You can also use a Function's left grab handle to drag and drop it into, or out of, a group. A saved group that's empty displays a dashed target into which you can drag and drop Functions.


Group Name*: Groupy McGroupface

Drag-and-drop target

;

6.1. 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,u
ntrusted,trusted,ethernet1/3,ethernet1/2,log-forwarding-default,2018/09/20
13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allow,296,296,0,4,2018/09/20
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-
out,0,0,0,0,,PA-VM,from-policy,,,0,,0,,N/A,0,0,0,0
```

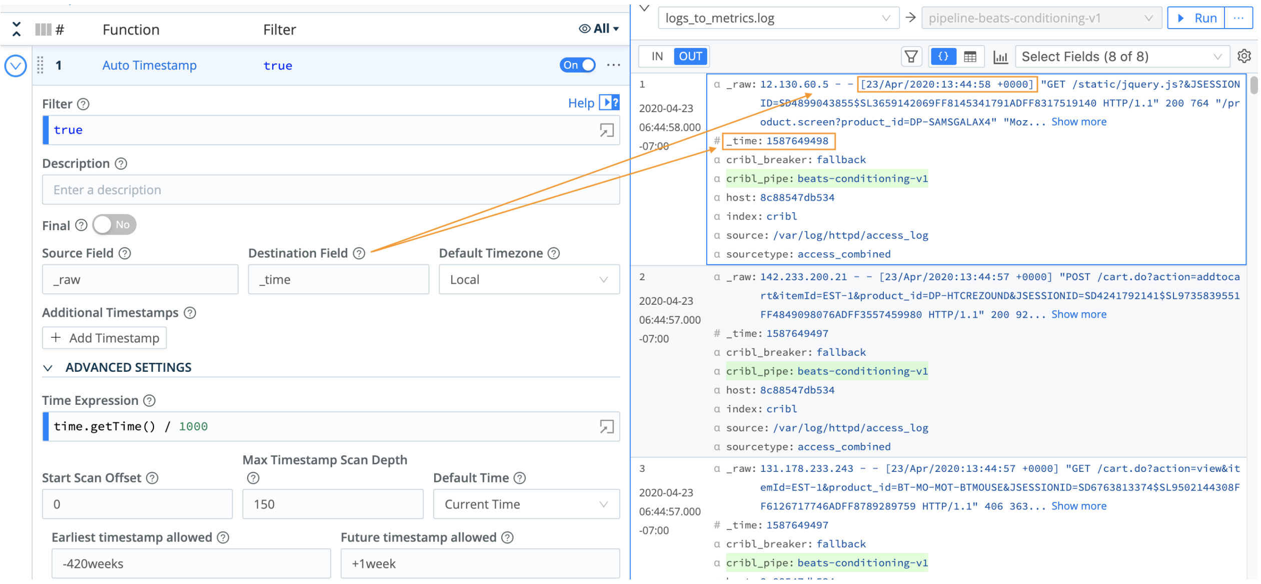
To add this sample (after creating an Auto Timestamp Function with the above **Filter** expression): Go to **Preview** > **Add a Sample** > **Paste a Sample**, and add the data snippet above. Do not make any changes to timestamping or line breaking, and select **Save as Sample File**.

By default, Cribl 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 1569006235`
- **GMT**: Friday, 20 September 2019, 7:03:55 PM GMT
- **Your Local Time**: Friday, 20 September 2019 PDT, 12:03:55 AM **GMT** -07:00

Because no explicit timezone has been set (under **Default Timezone**), `_time` will inherit the **Local** timezone, which in this example is `GMT -07:00`.



Filter: true

Description: Enter a description

Final: No

Source Field: _raw

Destination Field: _time

Default Timezone: Local

Time Expression: `time.getTime() / 1000`

Start Scan Offset: 0

Max Timestamp Scan Depth: 150

Default Time: Current Time

Earliest timestamp allowed: -420weeks

Future timestamp allowed: +1week

```
1              _raw: 12.130.60.5 - - [23/Apr/2020:13:44:58 +0000] "GET /static/jquery.js?&JSESSION
2020-04-23           ID=SD4899043855$SL3659142069FF8145341791ADFF8317519140 HTTP/1.1" 200 764 "/pr
06:44:58.000         oduct.screen?product_id=DP-SAMSGALAX4" "Moz... Show more
-07:00         _time: 1587649498
               cribl_breaker: fallback
               cribl_pipe: beats-conditioning-v1
               host: 8c88547db534
               index: cribl
               source: /var/log/httpd/access_log
               sourcetype: access_combined
2              _raw: 142.233.200.21 - - [23/Apr/2020:13:44:57 +0000] "POST /cart.do?action=addtoca
2020-04-23           rt&itemId=EST-1&product_id=DP-HTCREZOUND&JSESSIONID=SD4241792141$SL9735839551
06:44:57.000         FF4849098076ADFF3557459980 HTTP/1.1" 200 92... Show more
-07:00         _time: 1587649497
               cribl_breaker: fallback
               cribl_pipe: beats-conditioning-v1
               host: 8c88547db534
               index: cribl
               source: /var/log/httpd/access_log
               sourcetype: access_combined
3              _raw: 131.178.233.243 - - [23/Apr/2020:13:44:57 +0000] "GET /cart.do?action=view&it
2020-04-23           emId=EST-1&product_id=BT-MO-MOT-BTMOUSE&JSESSIONID=SD6763813374$SL9502144308F
06:44:57.000         F6126717746ADFF8789289759 HTTP/1.1" 406 363... Show more
-07:00         _time: 1587649497
               cribl_breaker: fallback
               cribl_pipe: beats-conditioning-v1
```

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, but Cribl Stream will truncate it to millisecond resolution.

For further examples, see Extracting Timestamps from Messy Logs.

;

6.2. Aggregations

The Aggregations Function performs aggregate statistics on event data.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Time window: The time span of the tumbling window for aggregating events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Aggregates: Aggregate function(s) to perform on events.
E.g., `sum(bytes).where(action=='REJECT').as(TotalBytes)`. Expression format: `aggFunction(<FieldExpression>).where(<FilterExpression>).as(<outputField>)`. See more examples below.

- Note: When used without `as()`, the aggregate's output will be placed in a field labeled `<aggFunction>_<fieldName>`. If there are conflicts, the last aggregate wins. For example, given two aggregates – `sum(bytes).where(action=='REJECT')` and `sum(bytes)` – the latter one (`sum_bytes`) is the winner.

Group by Fields: Fields to group aggregates by. Supports wildcard expressions.

Evaluate fields: Set of key/value pairs to evaluate and add/set. Fields are added in the context of an aggregated event, before they're sent out. Does not apply to passthrough events.

Time Window Settings

Cumulative aggregations: If enabled, aggregations will be retained for cumulative aggregations when flushing out an aggregation table event. When set to `No` (the default), aggregations will be reset to `0` on flush.

Lag tolerance: The lag tolerance represents the tumbling window tolerance to late events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Idle bucket time limit: The amount of time to wait before flushing a bucket that has not received events. Must be a valid time string (e.g., `10s`). Must match pattern `\d+[sm]$`.

Output Settings

Passthrough mode : Determines whether to pass through the original events along with the aggregation events. Defaults to `No`.

Metrics mode: Determines whether to output aggregates as metrics. Defaults to `No`, causing aggregates to be output as events.

Sufficient stats mode: Determines whether to output *only* statistics sufficient for the supplied aggregations. Defaults to `No`, meaning output richer statistics.

Output prefix: A prefix that is prepended to all of the fields output by this Aggregations Function.

Advanced Settings

Aggregation event limit: The maximum number of events to include in any given aggregation event. Defaults to unlimited. Must be at least `1`.

Aggregation memory limit: The memory usage limit to impose upon aggregations. Defaults to unlimited (i.e., the amount of memory available in the system). Accepts numerals with multiple-byte units, like KB, MB, GB, etc. (such: as `4GB`.)

Flush on stream close: If set to `Yes` (the default), aggregations will flush when an input stream is closed. If set to `No`, the Time Window Settings will control flush behavior; this can be preferable in cases like the following:

- Your input data consists of many small files.
- You are sending data to Prometheus. Enabling **Flush on stream close** can send Prometheus multiple aggregations from the same Worker Process for the same time period. Prometheus cannot tell the multiple aggregations apart, and will ingest only the first one.

List of Aggregate Functions

- `avg(expr:FieldExpression)`: Returns the average of the values of the parameter.

- `count(expr:FieldExpression)`: Returns the number of occurrences of the values of the parameter.

- `dc(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of the `<expr>` parameter, within a relative error rate.

- `distinct_count(expr: FieldExpression, errorRate: number = 0.01)`: Returns the estimated number of distinct values of the `<expr>` parameter, within a relative error rate.

- `earliest(expr:FieldExpression)`: Returns the earliest (based on `_time`) observed value of the parameter.

- `first(expr:FieldExpression)`: Returns the first observed value of the parameter.

- `last(expr:FieldExpression)`: Returns the last observed value of the parameter.

- `latest(expr:FieldExpression)`: Returns the latest (based on `_time`) observed value of the parameter.

- `list(expr:FieldExpression[,max:number])`: Returns a list of values of the parameter.

  - Optional `max` parameter limits the number of values returned. If omitted, the default is `100`. If set to `0`, will return all values.

- `max(expr:FieldExpression)`: Returns the maximum value of the parameter.

- `median(expr:FieldExpression)`: Returns the middle value of the sorted parameter.

- `min(expr:FieldExpression)`: Returns the minimum value of the parameter.

- `per_second(expr:FieldExpression)`: Returns the per second rate (based on `_time`) observed value of the parameter.

- `perc(level: number, expr: FieldExpression)`: Returns `<level>` percentile value of the numeric values of the `<expr>` parameter.

- `rate(expr:FieldExpression, timeString: string = '1s')`: Returns the rate (based on `_time`) observed value of the parameter.

- `stdev(expr:FieldExpression)`: Returns the sample standard deviation of the values of the parameter.

- `stdevp(expr:FieldExpression)`: Returns the population standard deviation of the values of the parameter.

- `sum(expr:FieldExpression)`: Returns the sum of the values of the parameter.

- `sumsq(expr:FieldExpression)`: Returns the sum of squares of the values of the parameter.
- `values(expr:FieldExpression[,max:number,errorRate:number])`: Returns a list of distinct values of the parameter.
    - Optional `max` parameter limits the number of values returned; if omitted, the default is `0`, meaning return all distinct values.
    - Optional `errorRate` parameter controls how accurately the function counts “distinct” values. Range is `0`–`1`; if omitted, the default value is `0.01`. Higher values allow higher error rates (fewer unique values recognized), with the offsetting benefit of less memory usage.
- `variance(expr:FieldExpression)`: Returns the sample variance of the values of the parameter.
- `variancep(expr:FieldExpression)`: Returns the population variance of the values of the parameter.

Safeguarding Data

Upon shutdown, Cribl 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.)

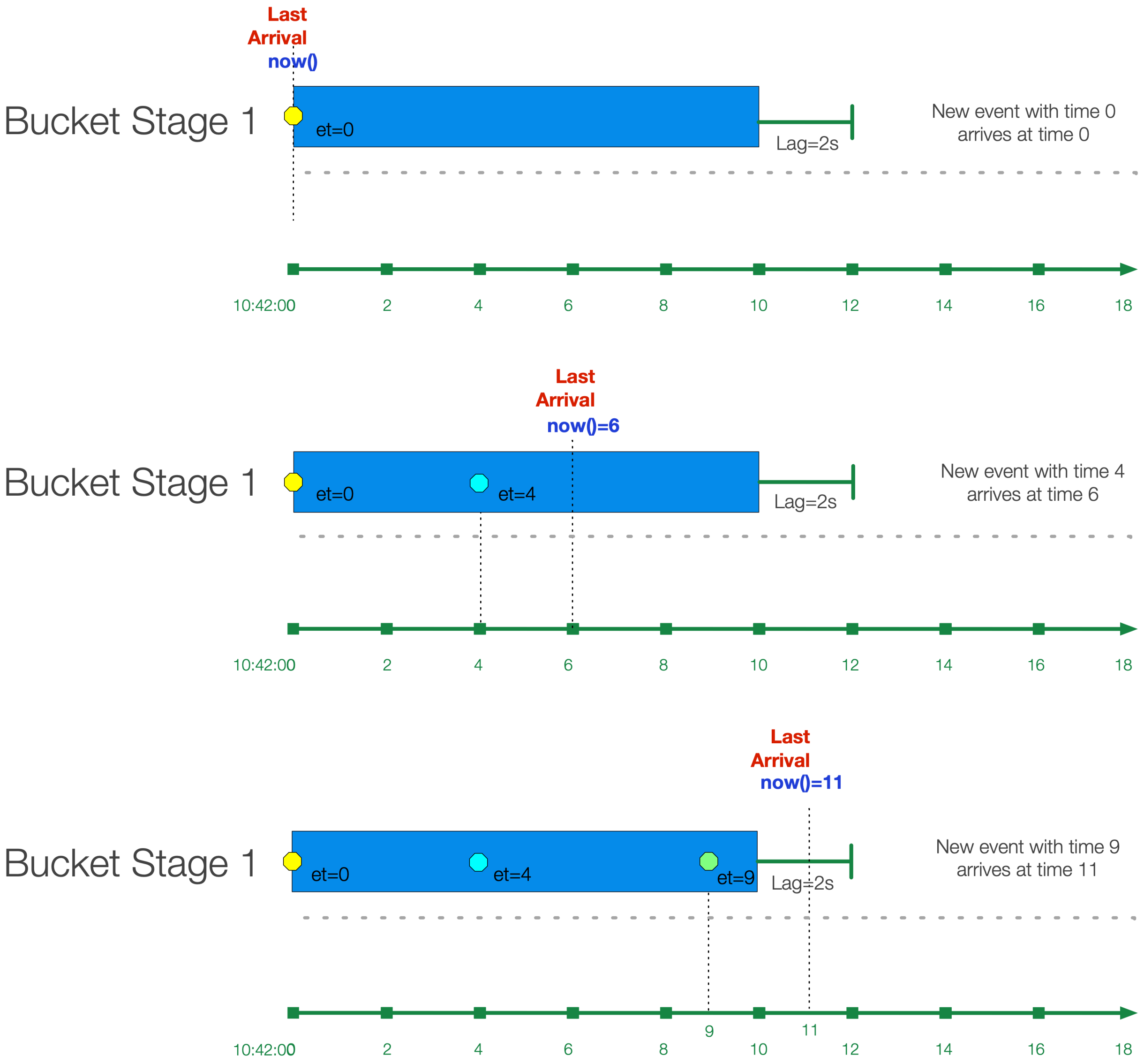
How Do Time Window Settings Work?

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.

10s Window Aggregation **Settings**: Lag=2s, IdleTime=4s



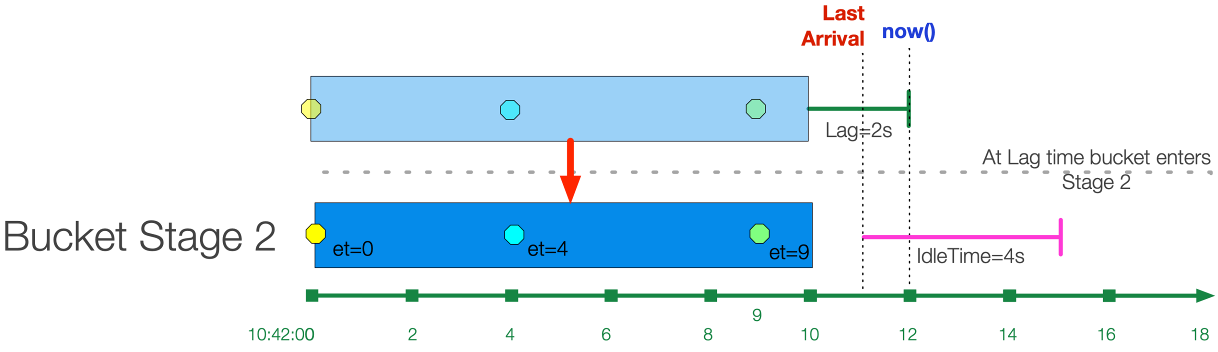
New event with time 0 arrives at time 0

New event with time 4 arrives at time 6

New event with time 9 arrives at time 11

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.

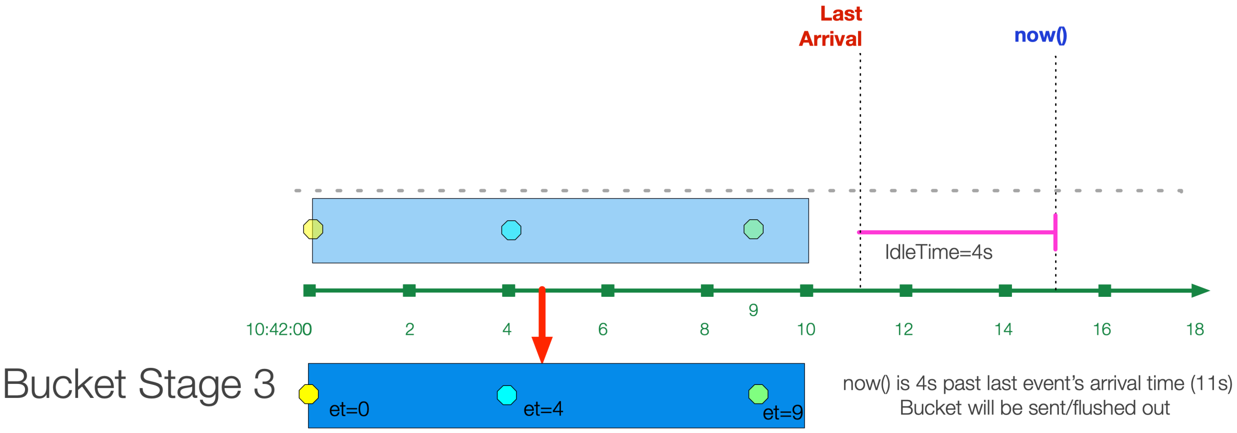
After the lag time expires, the bucket moves to Stage 2.



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.

> Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

;

6.3. 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

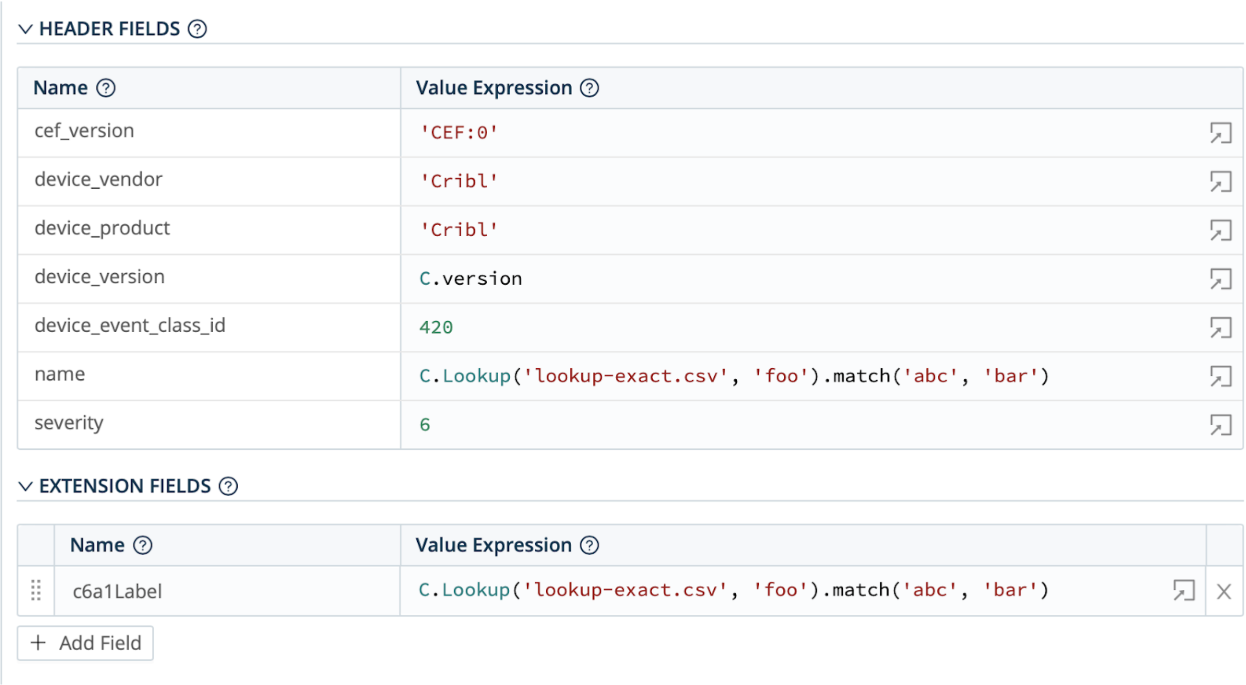
CEF Extension field definitions. Field names and values will be written in `key=value` format. Select each field's Name from the drop-down list. Values can be computed with JS expressions, or can be constants.

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
```

;

6.4. Chain

The Chain Function does one thing: It chains data processing from a Pipeline or Pack to another Pipeline or Pack. This can be useful for sequential processing, or just to separate groups of related Functions into discrete Pipeline or Pack units that make intuitive sense.

This Function includes guardrails against circular references. Still, Cribl recommends that you keep chained configurations understandable by all your users. There are also different scope restrictions when using Chain in a Pack versus in a Pipeline:

- 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.

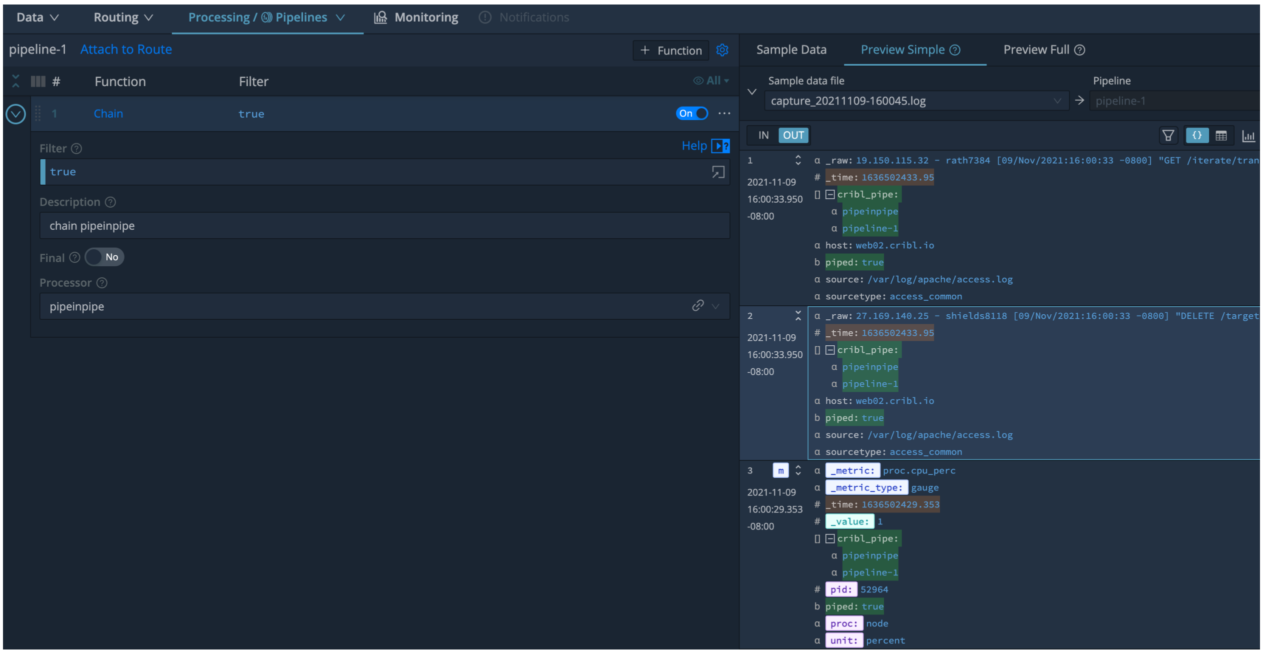
Description: Optionally, add a simple description of this Function's purpose. Defaults to empty.

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.



pipeline-1

#	Function	Filter	
1	Chain	true	On

Filter: true

Description: chain pipeinpipe

Final: No

Processor: pipeinpipe

Sample data file: capture_20211109-160045.log → Pipeline: pipeline-1

```
1    2021-11-09 16:00:33.950 -08:00
  α _raw: 19.150.115.32 - rath7384 [09/Nov/2021:16:00:33 -0800] "GET /iterate/tran
  # _time: 1636502433.95
  [] cribl_pipe:
      α pipeinpipe
      α pipeline-1
  α host: web02.cribl.io
  b piped: true
  α source: /var/log/apache/access.log
  α sourcetype: access_common
2    2021-11-09 16:00:33.950 -08:00
  α _raw: 27.169.140.25 - shields8118 [09/Nov/2021:16:00:33 -0800] "DELETE /target
  # _time: 1636502433.95
  [] cribl_pipe:
      α pipeinpipe
      α pipeline-1
  α host: web02.cribl.io
  b piped: true
  α source: /var/log/apache/access.log
  α sourcetype: access_common
3  m 2021-11-09 16:00:29.353 -08:00
  α _metric: proc.cpu_perc
  α _metric_type: gauge
  # _time: 1636502429.353
  # _value: 1
  [] cribl_pipe:
      α pipeinpipe
      α pipeline-1
  # pid: 52964
  b piped: true
  α proc: node
  α unit: percent
```

cribl_pipe field shows whole processing path

;

6.5. 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.

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.

;

6.6. 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`, and `window`.

Code Functions **can** include `for` loops, `while` loops, and JavaScript methods such as `map`, `reduce`, `forEach`, `some`, and `every`. For further details, see Supported JavaScript Options.

Cribl 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`.

Notes and Examples

Functions (including the Code Function) always use the special variable `__e` to access the `(context)` event inside JavaScript expressions.

Possibly the simplest Code Function creates a new field and then assigns it a value:

```
__e['foo'] = 'Hello, Goats!'
```

For more ambitious implementations, see Code Function Examples.

Supported JavaScript Options

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

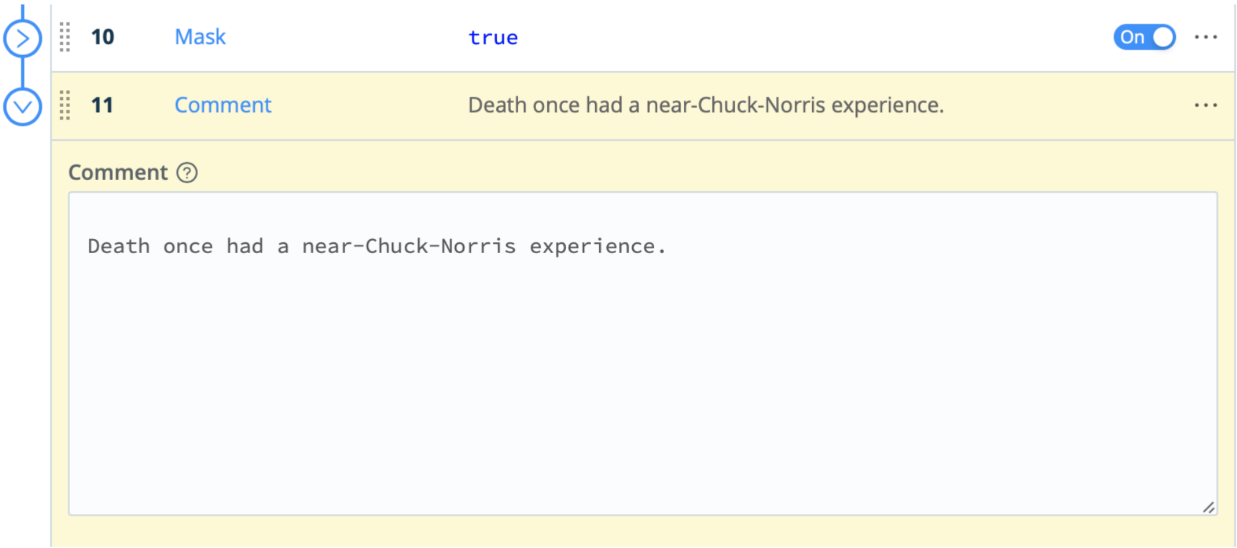
;

6.7. Comment

The Comment Function adds a text comment in a Pipeline. It makes no changes to event data. The added comment is visible only within the Pipeline UI, where it is useful for labeling Pipeline steps.

Usage

Comment: Add your comment as plain text in this field.

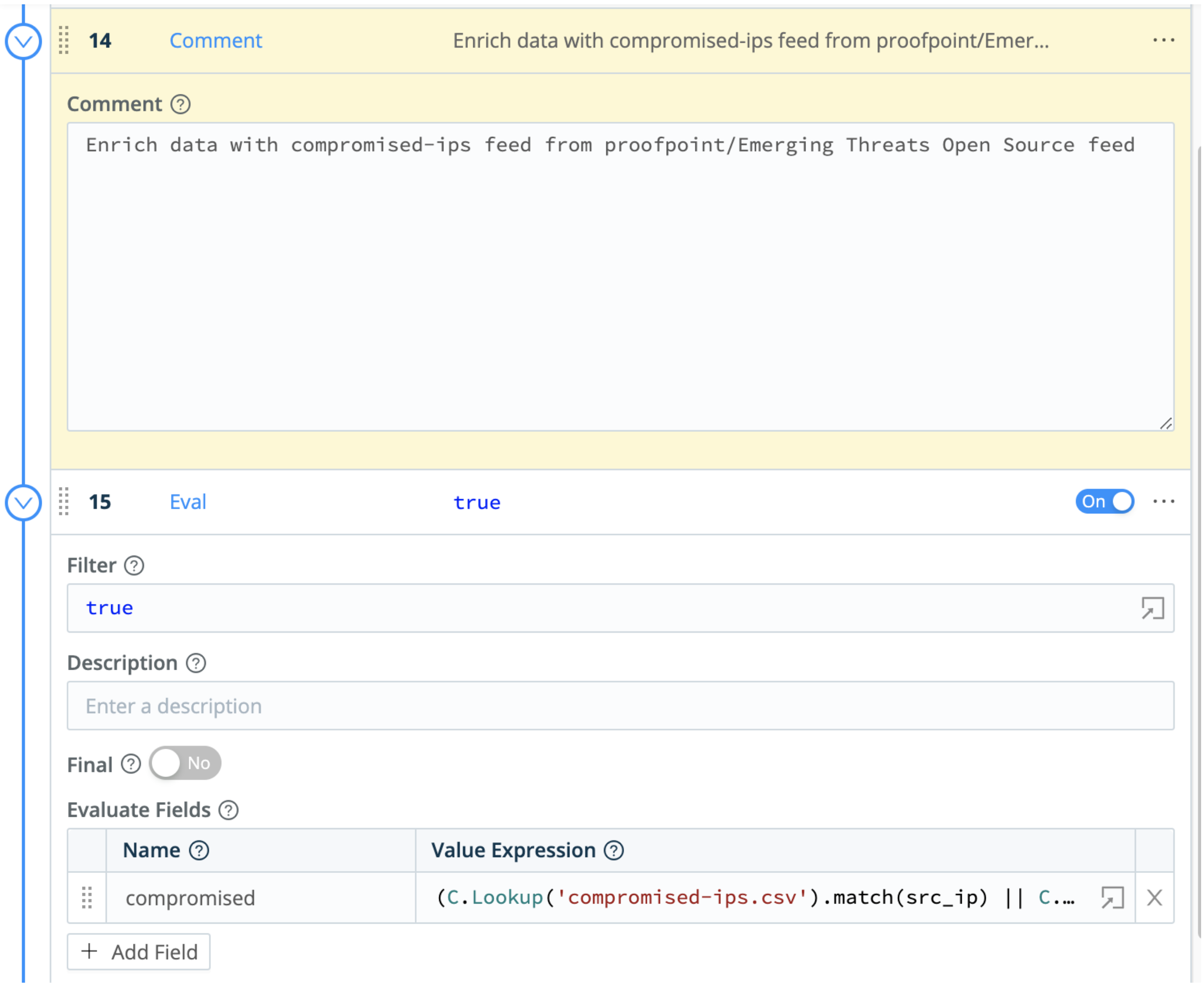


Comment

```
Death once had a near-Chuck-Norris experience.
```

Examples

This comment labels the Pipeline's next function:



14 Comment Enrich data with compromised-ips feed from proofpoint/Emer...

Comment

```
Enrich data with compromised-ips feed from proofpoint/Emerging Threats Open Source feed
```

15 Eval true On

Filter

```
true
```

Description

Enter a description

Final No

Evaluate Fields

	Name	Value Expression
	compromised	`(C.Lookup('compromised-ips.csv').match(src_ip) || C.…`

;

6.8. 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 Cribl Stream's existing Reverse DNS Function, which is now deprecated.)

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**.

Advanced Settings

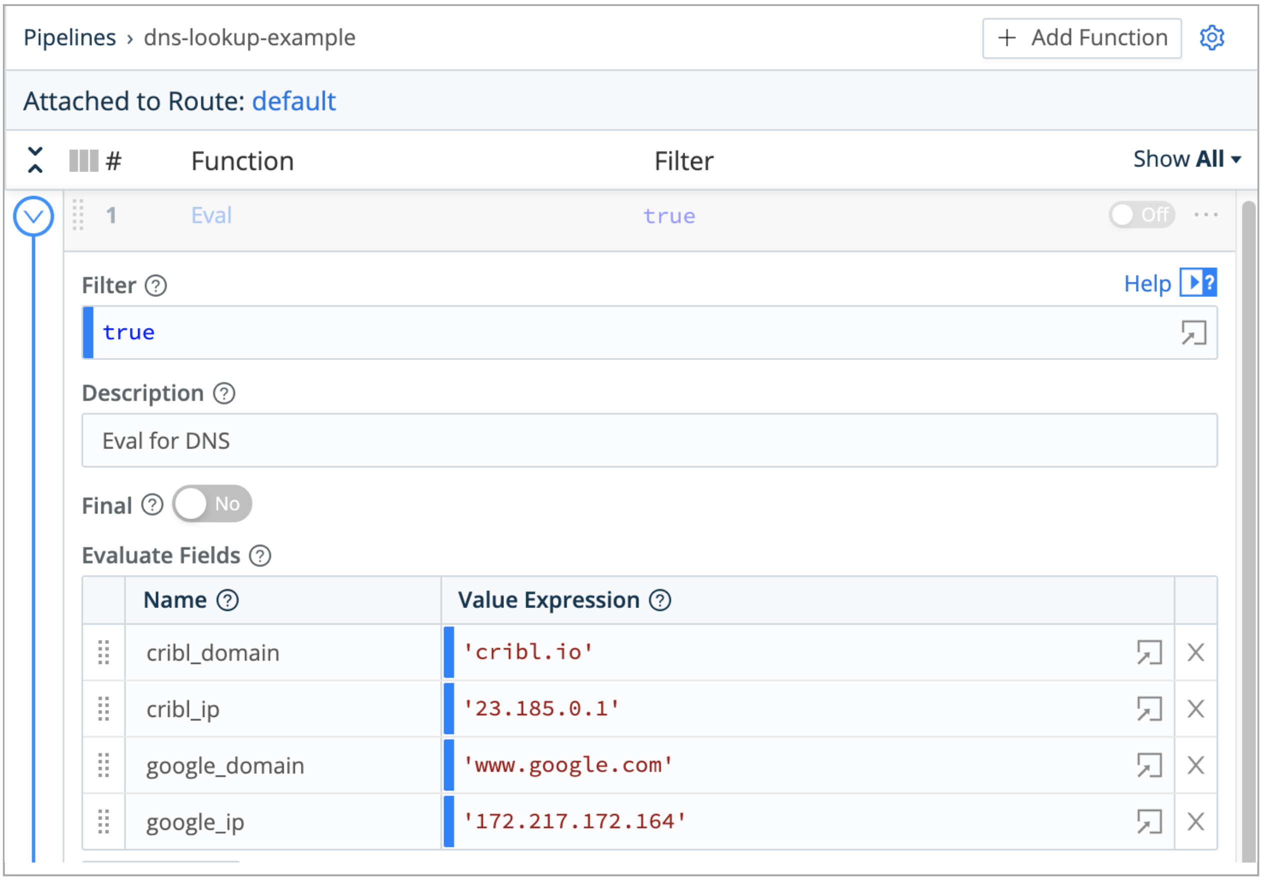
DNS server(s) overrides: IP address(es), in RFC 5952 format, of the DNS server(s) to use for resolution. IPv4 examples: `1.1.1.1`, `4.2.2.2:53`. IPv6 examples: `[2001:4860:4860::8888]`, `[2001:4860:4860::8888]:1053`. If this field is not specified, Cribl Stream will use the system's DNS server.

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`, contact Cribl Support to understand the implications. Highest allowed value is `10000`.

Example

This example Pipeline chains two Functions. First, we have an **Eval** Function that defines key-value pairs for two alphabetical domain names and two numeric IP addresses.


Pipelines › dns-lookup-example
+ Add Function
Attached to Route: default
# Function Filter Show All
Off
Help

1 Eval true

Filter: true

Description: Eval for DNS

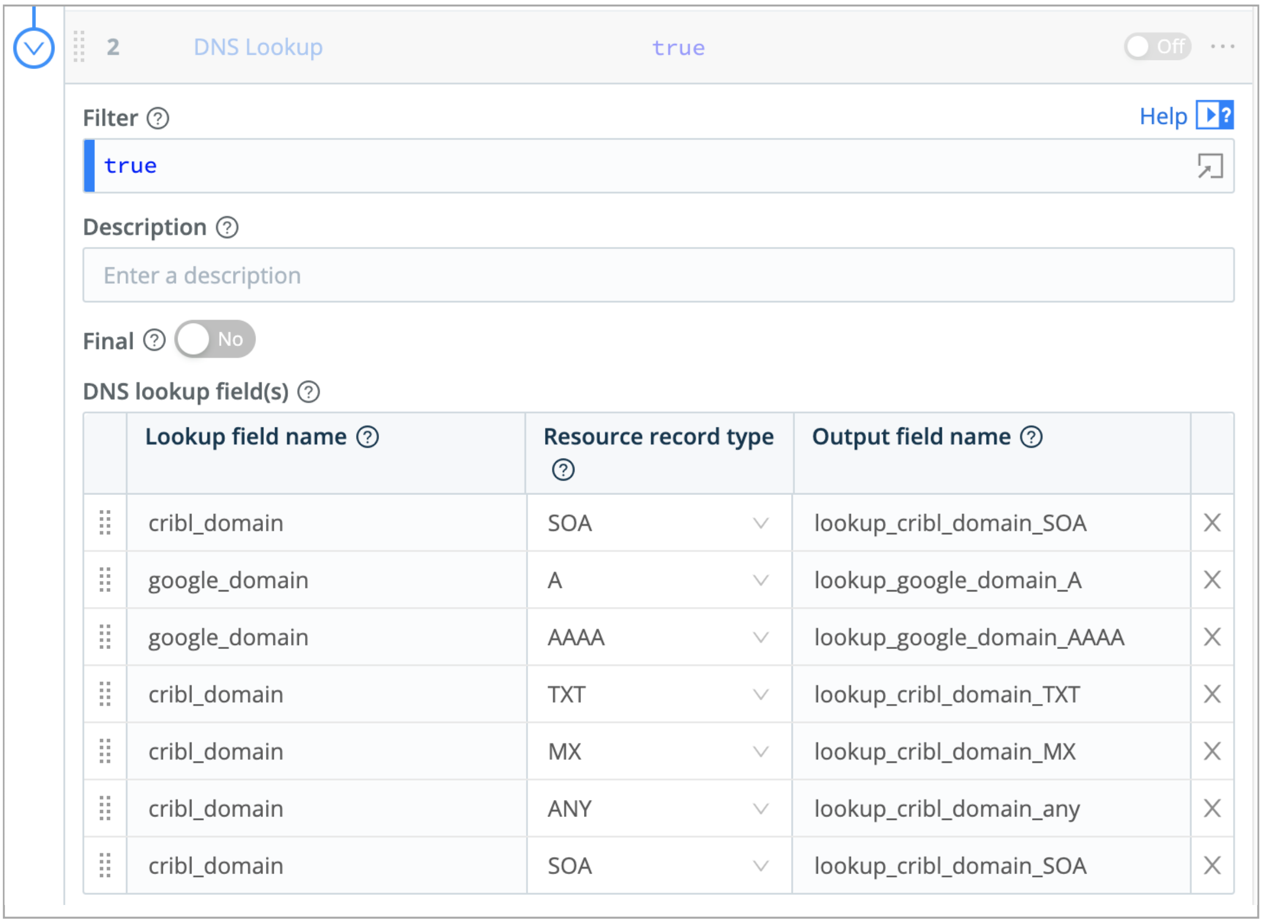
Final: No

Evaluate Fields

Name	Value Expression
cribl_domain	'cribl.io'
cribl_ip	'23.185.0.1'
google_domain	'www.google.com'
google_ip	'172.217.172.164'

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.



Filter: true

Description: Enter a description

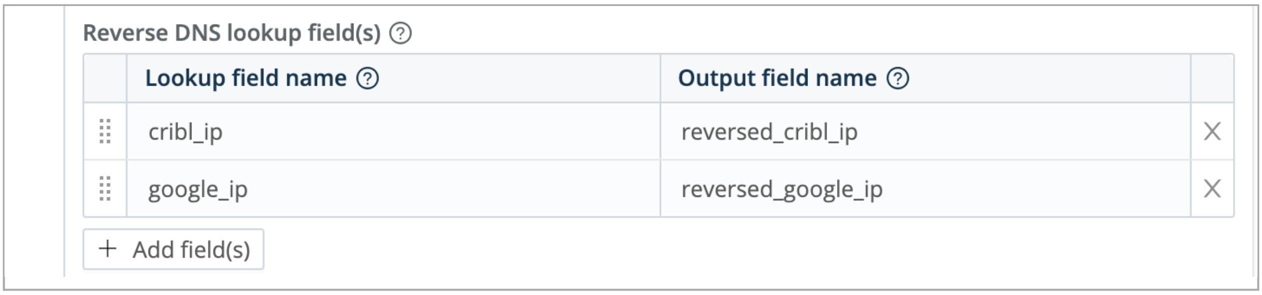
Final: No

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	TXT	lookup_cribl_domain_TXT
cribl_domain	MX	lookup_cribl_domain_MX
cribl_domain	ANY	lookup_cribl_domain_any
cribl_domain	SOA	lookup_cribl_domain_SOA

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

;

6.9. 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'`

;

6.10. 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.

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 select a sample rate a priori, 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

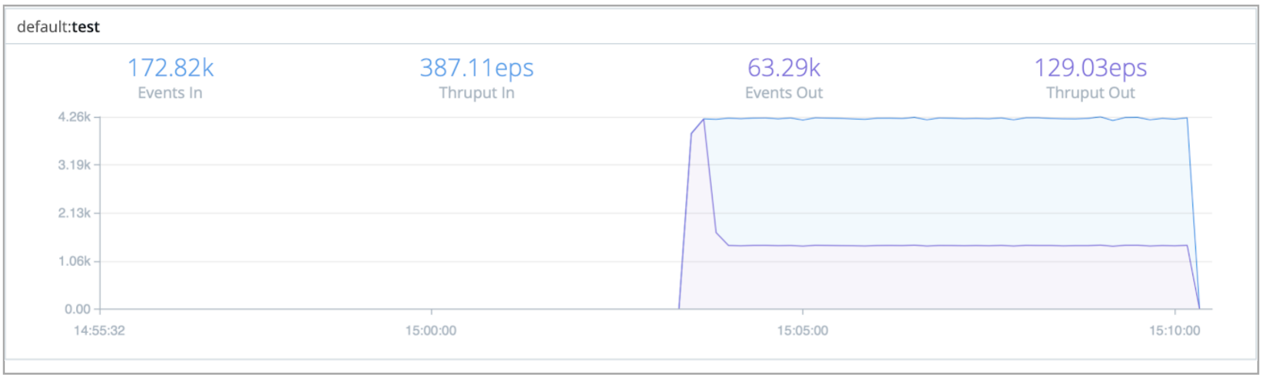
Here's an example that illustrates the effectiveness of using the Square Root sample mode.

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.
>
> Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

;

6.11. 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: Set of key/value pairs to add. The left-hand side input (**Name**) is the key name. The right-hand side input (**Value Expression**) is a JS expression to compute the value – this can be a constant. Nested addressing is supported. Strings intended to be used as values must be single- or double-quoted. (For details, see Cribl Expression Syntax.)

Keep fields: List of fields to keep. Wildcards (*) and nested addressing are supported. Takes precedence over **Remove fields** (below). To reference a parent object and all children requires using 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. Wildcards (*) and nested addressing are supported. Cannot remove fields matching **Keep fields**. Cribl Stream internal fields that start with `__` (double underscore) **cannot** be removed via wildcard. Instead, they need to be specified 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.

;

6.12. 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.

> 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.

;

6.13. 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.

;

6.14. 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`

NAME	VALUE EXPRESSION
`Country`	`__geoip.country.names.en`
`Zip`	`__geoip.postal.code`
`Lat`	`__geoip.location.latitude`
`Long`	`__geoip.location.longitude`

In the Eval Function's **Remove fields** setting, you could specify the `__geoip` field for removal, if desired. However, its `__` prefix makes it an internal field anyway.

> For a hosted tutorial on applying the GeoIP Function, see Cribl's GeoIP and Threat Feed Enrichment Sandbox.

;

6.15. 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 links for creating and testing Grok patterns: http://grokdebug.herokuapp.com and http://grokconstructor.appspot.com/.
- Additional patterns are available here: https://github.com/logstash-plugins/logstash-patterns-core/tree/master/patterns.

;

6.16. JSON Unroll

The JSON Unroll Function accepts a JSON object string `_raw` field, unrolls/explodes an **array** of **objects** therein into individual events, while also inheriting top level fields. See example(s). Cribl highly recommends not using this JSON Unroll function for certain types of data. Instead, perform the unrolling using an event breaker for those inputs which support configuring an event breaker. Specifying the event breaker type **JSON Array** and toggling the **JSON Extract Fields** option to **Yes** will accomplish the same unrolling but much more efficiently. This is recommended, for example, for CloudTrail and Office635 events, which are collected as JSON arrays.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Path: Path to array to unroll, e.g., `foo.0.bar`.

New name: The name that the exploded array element will receive in each new event. Leave empty to expand the array element with its original name.

Example(s)

Assume you have an incoming event that has a `_raw` field as a JSON object string like this:

Sample _raw field

```
{"date":"9/25/18 9:10:13.000 PM",
   "name":"Amrit",
   "age":42,
   "allCars": [
       { "name":"Ford", "models":[ "Fiesta", "Focus", "Mustang" ] },
       { "name":"GM", "models":[ "Trans AM", "Oldsmobile", "Cadillac" ] },
       { "name":"Fiat", "models":[ "500", "Panda" ] },
       { "name":"Blackberry", "models":[ "KEY2", "Bold Touch 9900" ] }
   ]
}
```

Settings:

Path: `allCars` **New Name**: `cars`

Output Events:

Resulting Events

```
Event 1:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"Ford","models":["Fiesta","Focus","Mustang"]}}"}

Event 2:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"GM","models":["Trans AM","Oldsmobile","Cadillac"]}}"}

Event 3:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"Fiat","models":["500","Panda"]}}"}

Event 4:
{"_raw":"{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":
{"name":"Blackberry","models":["KEY2","Bold Touch 9900"]}}"}
```

Each element under the original **allCars** array is now placed in a **cars** field in its own event, inheriting original top level fields; **date**, **name** and **age**

;

6.17. 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 Libary, 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 mode**s, this attribute refines how to resolve multiple matches. `First match` will return the first matching entry. `Most specific` will scan all entries, finding the most specific match. `All` will return all matches in the output, as arrays. (Defaults to `First match`.
Not displayed for Exact **Match mode**.)

Lookup fields (.csv): Field(s) that should be used to key into the lookup table.

- **Lookup field name in event**: Exact field name as it appears in events. Nested addressing supported.
- **Corresponding field name in lookup**: The field name as it appears in the lookup file. Defaults to the **Lookup field name in event** value. This input is optional.

> ### Case-Sensitive / Multiple Matches
>
> Lookups are case-sensitive by default. (See the **Ignore case** option below.)
>
> If the lookup file contains duplicate key names with different values, all **Match mode**s of this Function will use **only** the value in the key's **final** instance, ignoring all preceding instances.

Output field(s): Field(s) to add to events after matching the lookup table. Defaults to **all** if not specified.

- **Output field name from lookup**: Field name, as it appears in the lookup file.
- **Lookup field name in event**: Field name to add to event. Defaults to the lookup field name. This input is optional. Nested addressing is supported.

Advanced Settings

Reload period (sec): Periodically check the underlying file for modtime changes, and reload if necessary. Use `-1` to disable. Defaults to `60`.

Ignore case: Ignore case when performing **Match mode: Exact** lookups. Defaults to `No`.

Add to raw event: Whether to append the looked-up values to the `_raw` field, as key=value pairs. Defaults to `No`.

Examples

Example 1: Regex Lookups

Assign a `sourcetype` field to events if their `_raw` field matches a particular regex.

```
regex,sourcetype
"^[^,]+,[^,]+,[^,]+,THREAT",pan:threat
"^[^,]+,[^,]+,[^,]+,TRAFFIC",pan:traffic
"^[^,]+,[^,]+,[^,]+,SYSTEM",pan:system
```

Match mode: Regex

Match type: First match

Lookup field name in event: `_raw`

Corresponding field name in lookup: `regex`

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/2
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/20
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,


### AFTER:

{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/2
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "sourcetype": "pan:traffic"
  }
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,FOOBAR,THREAT,end,2049,2018/09/20
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "sourcetype": "pan:threat"
  }
```

Example 2: CIDR Lookups

Assign a `location` field to events if their `destination_ip` field matches a particular CIDR range.

```
range,location
10.0.0.0/24,San Francisco
10.0.0.0/16,California
10.0.0.0/8,US
```

Match mode: CIDR

Match type: See options below

Lookup field name in event: `destination_ip`

Corresponding field name in lookup: `range`

In **Match mode: CIDR** with **Match type: Most specific**, the lookup will implicitly search for matches from most specific to least specific. There is no need to pre-sort data.

Note that **Match mode: CIDR** with **Match type: First Match** is likely the most performant with large lookups. This can be used as an alternative to **Most specific**, if the file is sorted with the most specific/relevant entries first. This mode still performs a table scan, top to bottom.

```
### BEFORE:

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "destination_ip": "10.0.0.102"
  }

### AFTER with Match Type: First Match

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "destination_ip": "10.0.0.102",
  "location": "San Francisco"
  }

### AFTER with Match Type: Most Specific

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "destination_ip": "10.0.0.102",
  "location": "San Francisco"
  }

### AFTER with Match Type: All

{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58,FOOBAR,TRAFFIC,end,2049,2018/09/
13:03:58,34.217.108.226,10.0.0.102,34.217.108.226,10.0.2.65,splunk,,,incomplete,vsys1,u
forwarding-default,2018/09/20 13:03:58,574326,1,53722,8088,53722,8088,0x400064,tcp,allo
13:03:45,7,any,0,730277,0x0,United States,10.0.0.0-10.255.255.255,0,4,0,aged-out,0,0,0,
  "destination_ip": "10.0.0.102",
  "location": [
    "San Francisco",
    "California",
    "US",
  ]}
```

More Examples and Scenarios

More examples:

- Ingest-time Lookups.
- Lookups and Regex Magic.
- Lookups as Filters for Masks.

See also:

- Managing Large Lookups to optimize file locations for large lookup files.
- Redis Function for faster lookups using a Redis integration.

;

6.18. 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: Simple description about this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Masking rules: Match Regex and Replace Expression pairs. Defaults to empty. Each row has the following fields:

- **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.

To add more rows, click **+ Add Rule**.

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"` ``

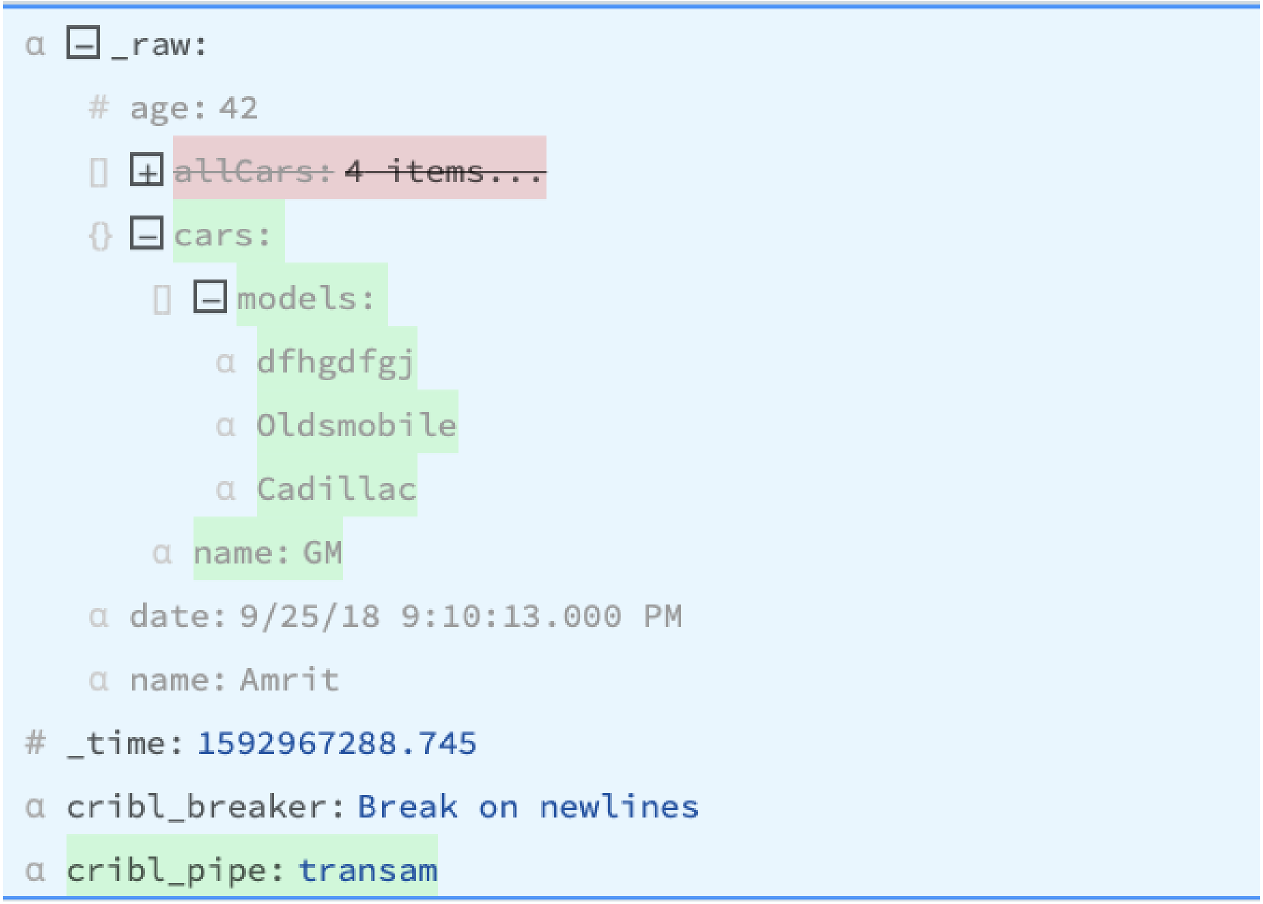
The **Replace expression** can reference other event fields as `event.<fieldName>`. For example, `` `${g1}${event.source}` ``. Note that this is slightly different from other expression inputs, where event fields are referenced without `event.` Here, we require the `event.` prefix for the following reasons:

- 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.

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:

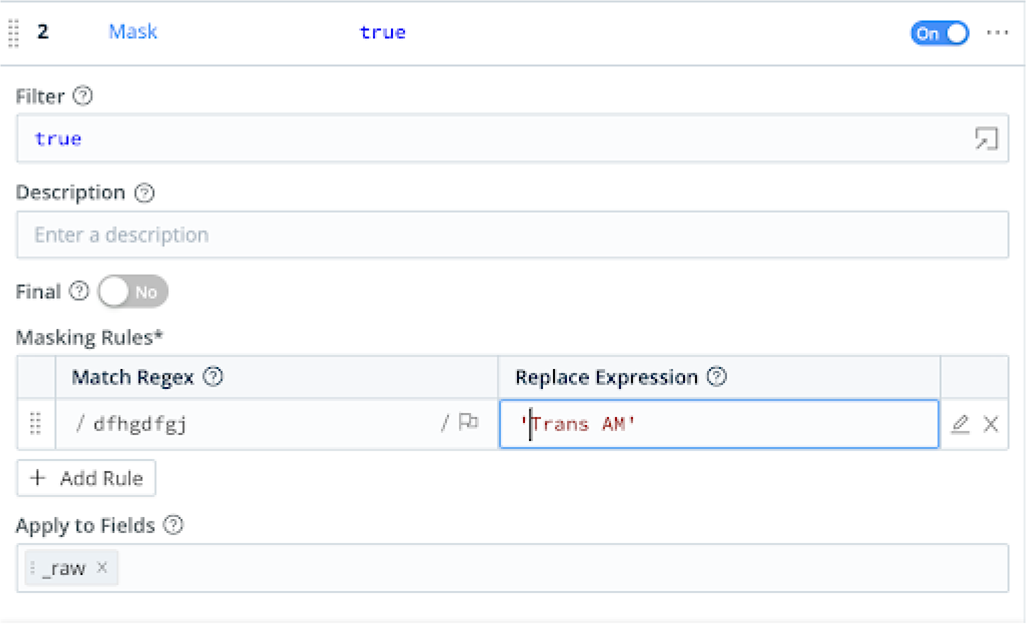


```
α ⊟ _raw:
    # age: 42
    [] ⊞ ~~allCars: 4 items...~~
    {} ⊟ cars:
        [] ⊟ models:
            α dfhgdfgj
            α Oldsmobile
            α Cadillac
        α name: GM
    α date: 9/25/18 9:10:13.000 PM
    α name: Amrit
# _time: 1592967288.745
α cribl_breaker: Break on newlines
α cribl_pipe: transam
```

Event before masking

Configure the Mask Function > Masking Rules as follows:

Match Regex: `dfhgdfgj` Replace Expression: `Trans AM`



2 Mask true On

Filter: `true`

Description: Enter a description

Final: No

Masking Rules*

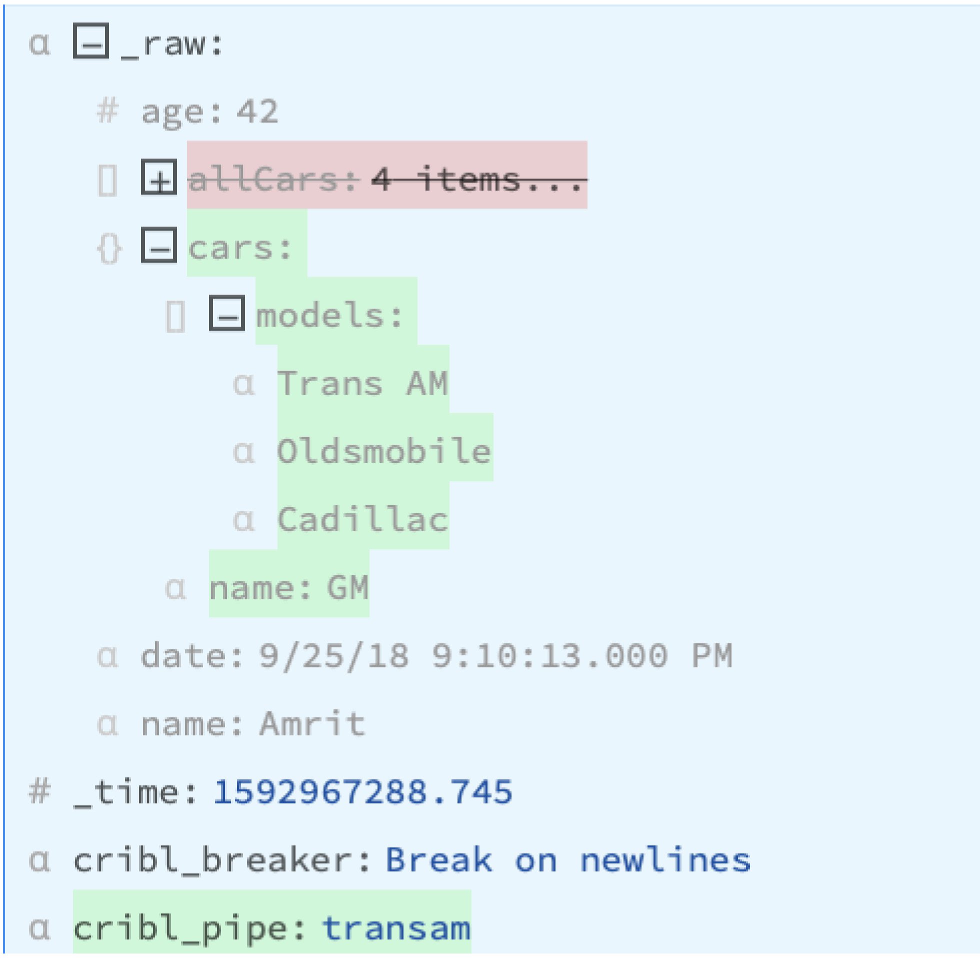
Match Regex	Replace Expression
/ dfhgdfgj /	'Trans AM'

+ Add Rule

Apply to Fields: _raw

Mask Function configuration

Result: Vroom vroom!

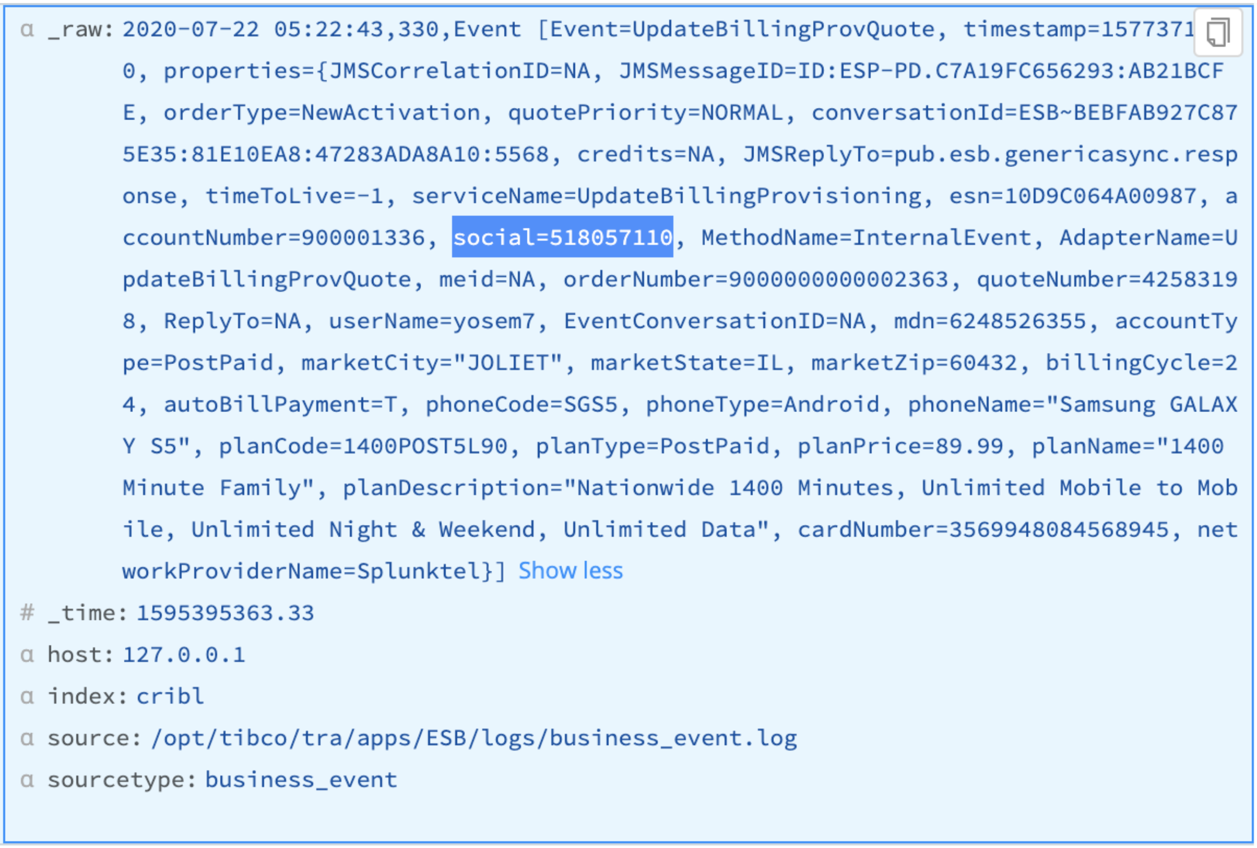


```
α ⊟ _raw:
    # age: 42
    [] ⊞ ~~allCars: 4 items...~~
    {} ⊟ cars:
        [] ⊟ models:
            α Trans AM
            α Oldsmobile
            α Cadillac
        α name: GM
    α date: 9/25/18 9:10:13.000 PM
    α name: Amrit
# _time: 1592967288.745
α cribl_breaker: Break on newlines
α cribl_pipe: transam
```

Event 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=#########`.



```
α _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
α host: 127.0.0.1
α index: cribl
α source: /opt/tibco/tra/apps/ESB/logs/business_event.log
α sourcetype: business_event
```

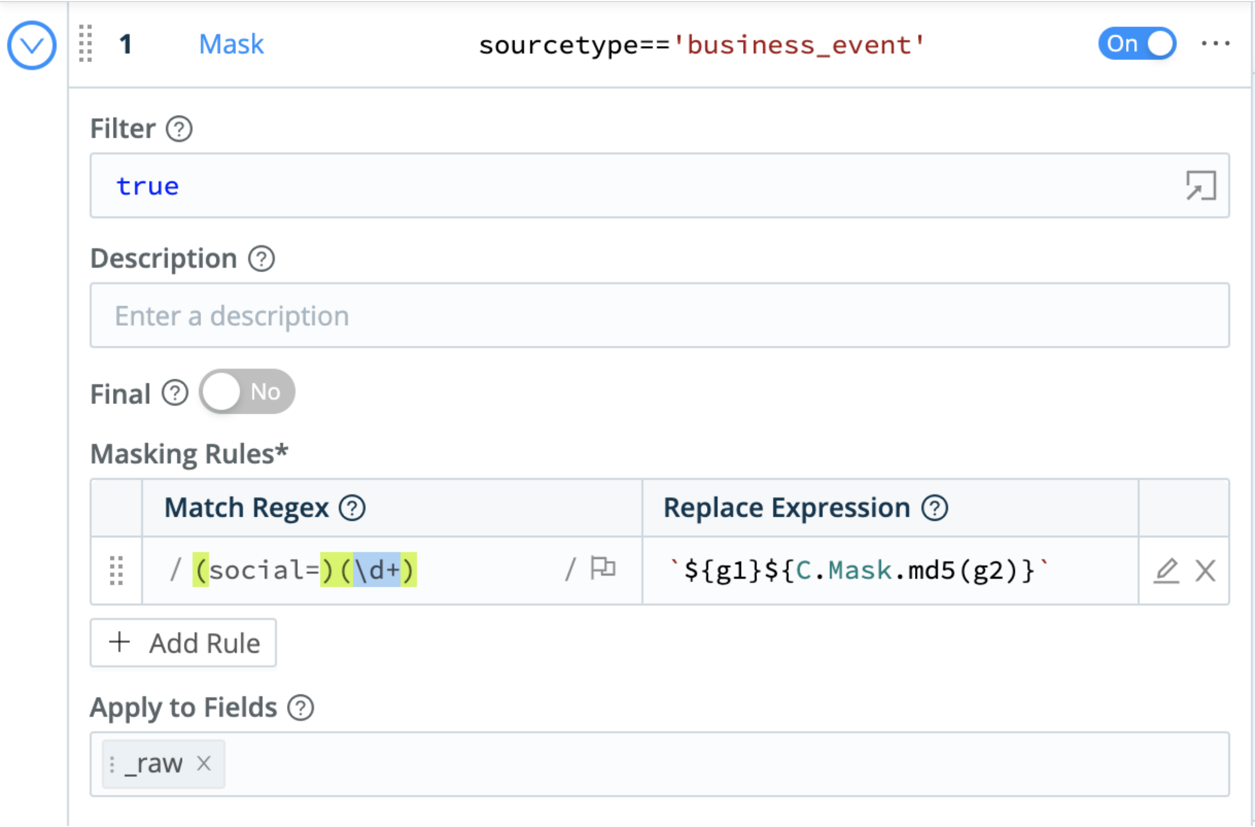
Event with unredacted SSNs

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.



1 Mask `sourcetype=='business_event'` On

Filter: `true`

Description: Enter a description

Final: No

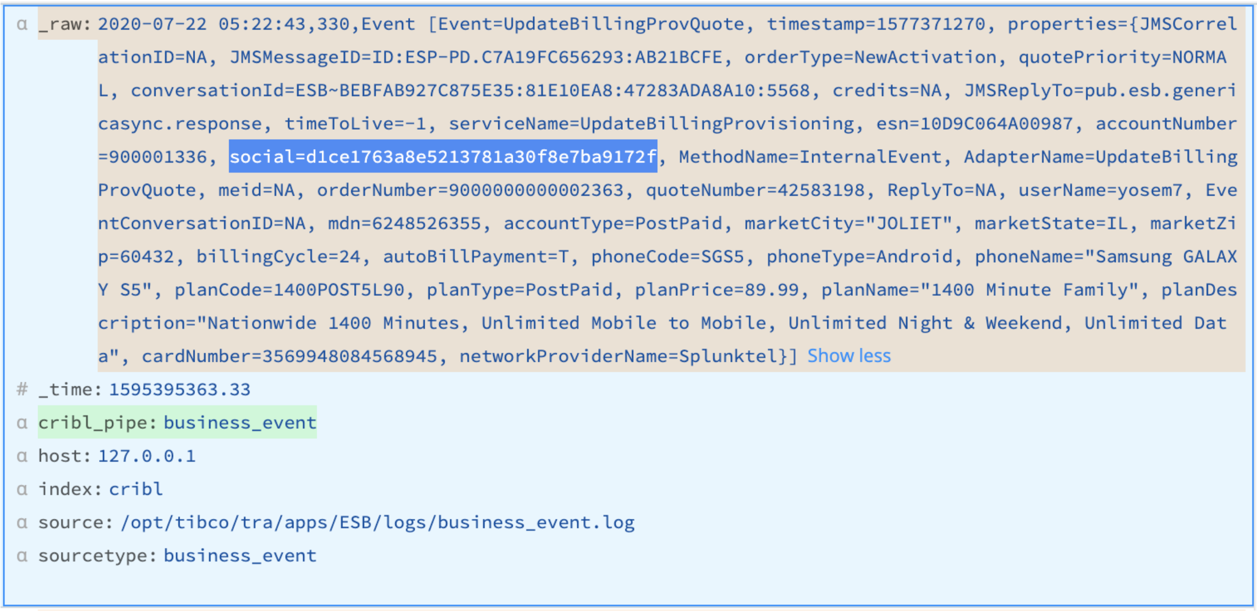
Masking Rules*

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

Apply to Fields: `_raw`

Mask Function configuration

Result: The sensitive values are replaced by their md5 hashes.



```
α _raw: 2020-07-22 05:22:43,330,Event [Event=UpdateBillingProvQuote, timestamp=1577371270, properties={JMSCorrel
        ationID=NA, JMSMessageID=ID:ESP-PD.C7A19FC656293:AB21BCFE, orderType=NewActivation, quotePriority=NORMA
        L, conversationId=ESB~BEBFAB927C875E35:81E10EA8:47283ADA8A10:5568, credits=NA, JMSReplyTo=pub.esb.generi
        casync.response, timeToLive=-1, serviceName=UpdateBillingProvisioning, esn=10D9C064A00987, accountNumber
        =900001336, social=d1ce1763a8e5213781a30f8e7ba9172f, MethodName=InternalEvent, AdapterName=UpdateBilling
        ProvQuote, meid=NA, orderNumber=9000000000002363, quoteNumber=42583198, ReplyTo=NA, userName=yosem7, Eve
        ntConversationID=NA, mdn=6248526355, accountType=PostPaid, marketCity="JOLIET", marketState=IL, marketZi
        p=60432, billingCycle=24, autoBillPayment=T, phoneCode=SGS5, phoneType=Android, phoneName="Samsung GALAX
        Y S5", planCode=1400POST5L90, planType=PostPaid, planPrice=89.99, planName="1400 Minute Family", planDes
        cription="Nationwide 1400 Minutes, Unlimited Mobile to Mobile, Unlimited Night & Weekend, Unlimited Dat
        a", cardNumber=3569948084568945, networkProviderName=Splunktel}] Show less
# _time: 1595395363.33
α cribl_pipe: business_event
α host: 127.0.0.1
α index: cribl
α source: /opt/tibco/tra/apps/ESB/logs/business_event.log
α 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
```

;

6.19. 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.

;

6.20. Parser

The Parser Function can be used to extract fields out of events, or to reserialize (rewrite) events with a subset of fields. Reserialization will maintain the format of the events.

For example: If an event contains comma-delimited fields, and `fieldA` and `fieldB` are filtered out, those fields' positions will be set to `null`, but not deleted completely.

Parser cannot remove fields that it did not create. A subsequent Eval Function can do so.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning 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`.

Operation mode: **Extract** will create new fields. **Reserialize** will extract, filter fields, and then reserialize.

Type: Parser/Formatter type to use. Options:

- CSV
- Extended Log File Format (ELFF)
- Common Log Format (CLF)
- K=V Pairs
- JSON
- Delimited Values

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: `-`

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 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 Serialize mode.

Destination field: Name of field in which to add extracted and serialized fields. If multiple new fields are created and this setting is configured then all new fields are created as elements of an array with the array name set to the name specified for this setting. If you want all new fields to be independent, rather than in an array, then specify them using **List of fields** below. (Extract and Serialize 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**. Nested addressing supported.

Fields to remove: List of fields to remove. Supports wildcards (`*`). Cannot remove fields matching **Fields to keep**. Nested addressing supported.

> Negated terms are supported in both **Fields to remove** and **Fields to keep**. When you use negated terms, the 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`."

Fields filter expression: Expression to evaluate against `{index, name, value}` context of each field. Return truthy to keep, falsy to remove field. Index is zero-based.

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.

How Fields Settings Interact

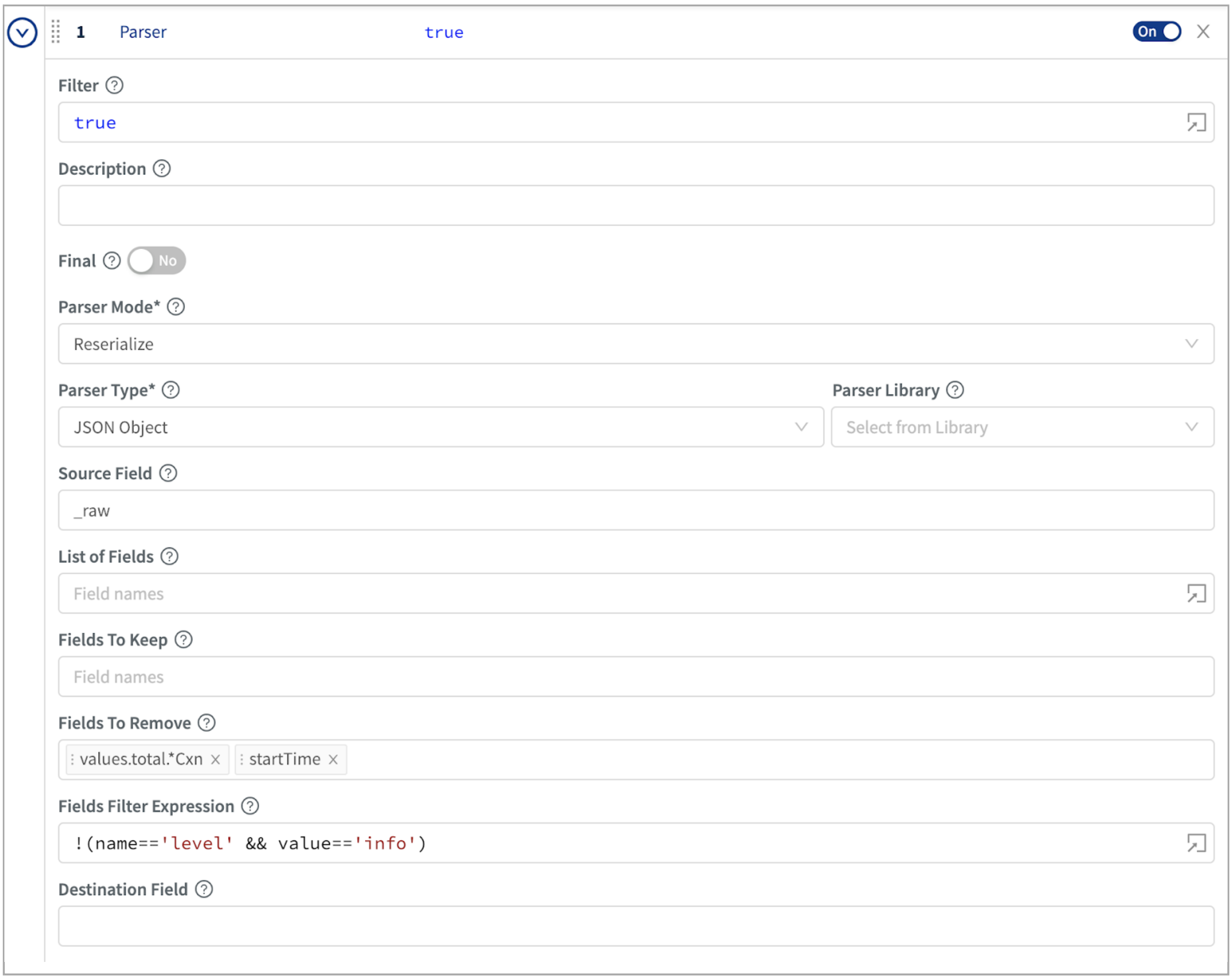
The **Fields to keep**, **Fields to remove**, and **Fields filter expression** settings interact as follows:

- Order of evaluation: **Fields to keep** > **Fields to remove** > **Fields filter expression**.
- 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.

Example 1

Insert the following sample, using **Preview** > **Add a Sample** > **Paste a Sample**: `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).



1 Parser true On

Filter: true

Description:

Final: No

Parser Mode*: Reserialize

Parser Type*: JSON Object

Parser Library: Select from Library

Source Field: _raw

List of Fields: Field names

Fields To Keep: Field names

Fields To Remove: values.total.*Cxn, startTime

Fields Filter Expression: `!(name=='level' && value=='info')`

Destination Field:

Parser Function initial configuration

First, set the **Parser type** to `Key=Value Pairs`.

Scenario A:

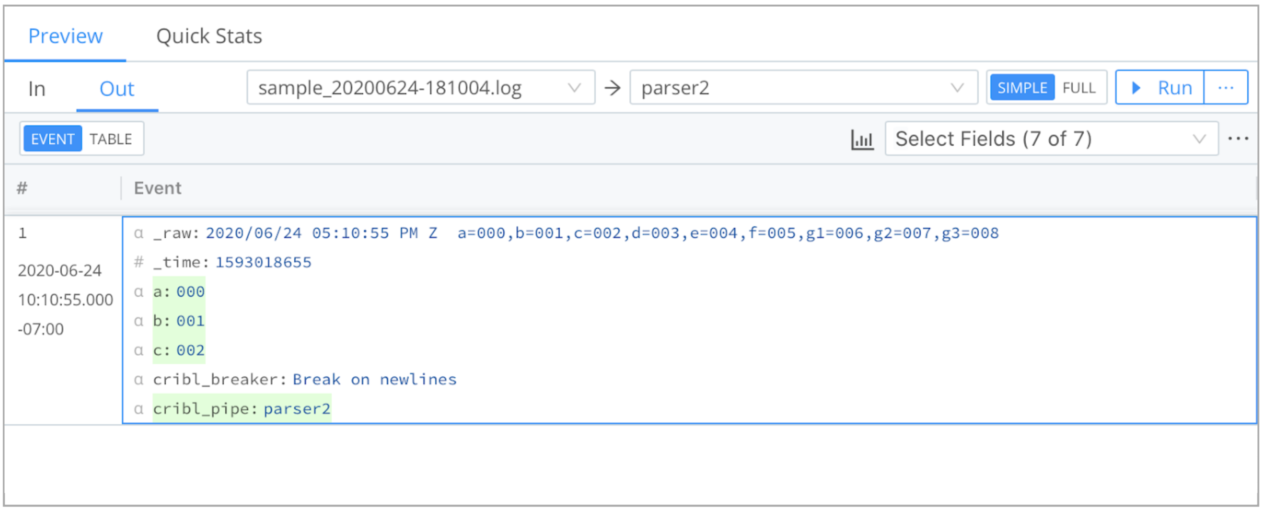
Keep fields `a`, `b`, `c`. Drop the rest.

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`



1
2020-06-24 10:10:55.000 -07:00

```
α _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
# _time: 1593018655
α a: 000
α b: 001
α c: 002
α cribl_breaker: Break on newlines
α 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`

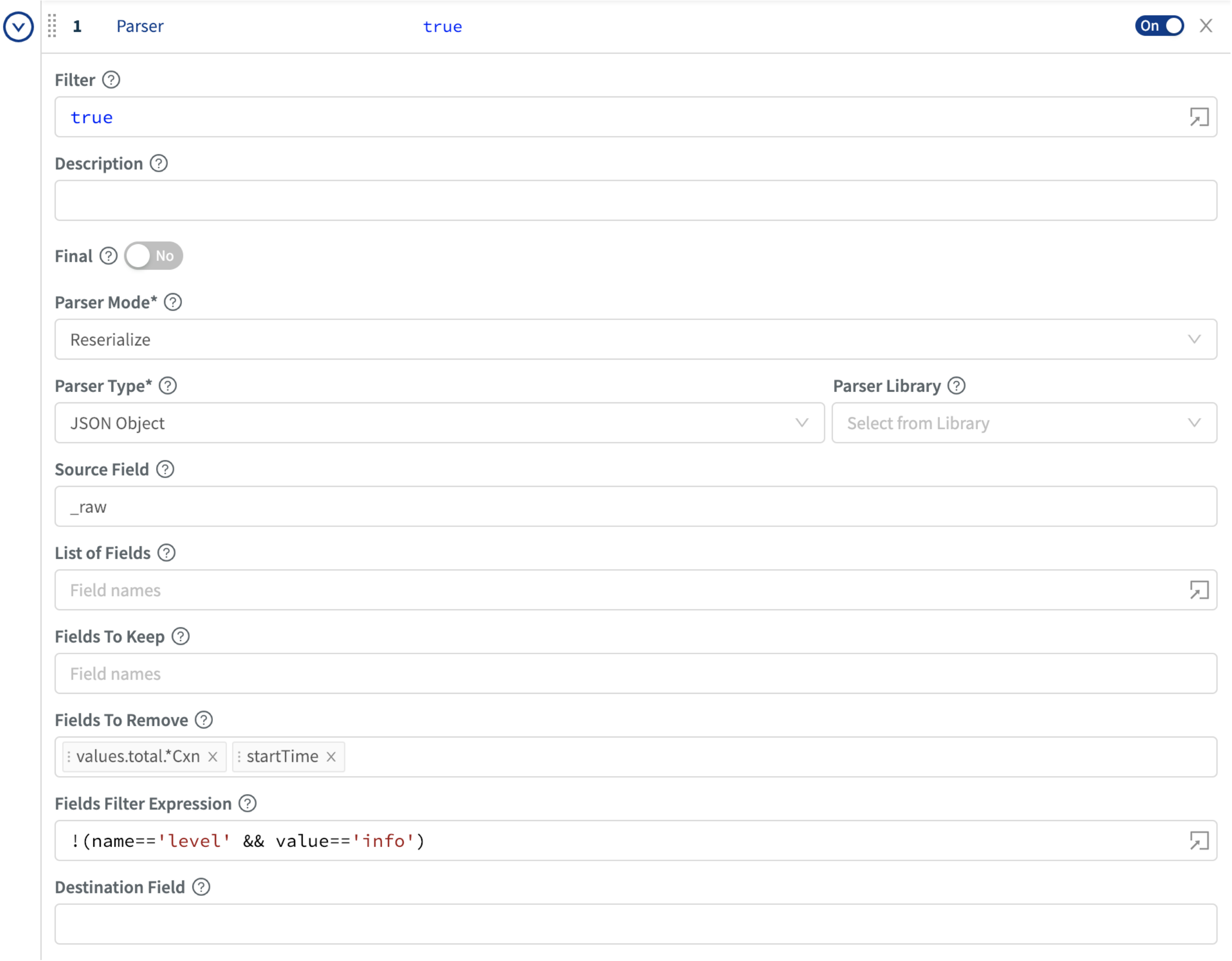
> The `index` refers to the location of a field in the array of all fields extracted by **this** Parser. It is zero-based. In the case above, `g2` and `g3` have `index` values of `7` and `8` , respectively.

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:



1 Parser true On

Filter: true

Description:

Final: No

Parser Mode*: Reserialize

Parser Type*: JSON Object

Parser Library: Select from Library

Source Field: _raw

List of Fields: Field names

Fields To Keep: Field names

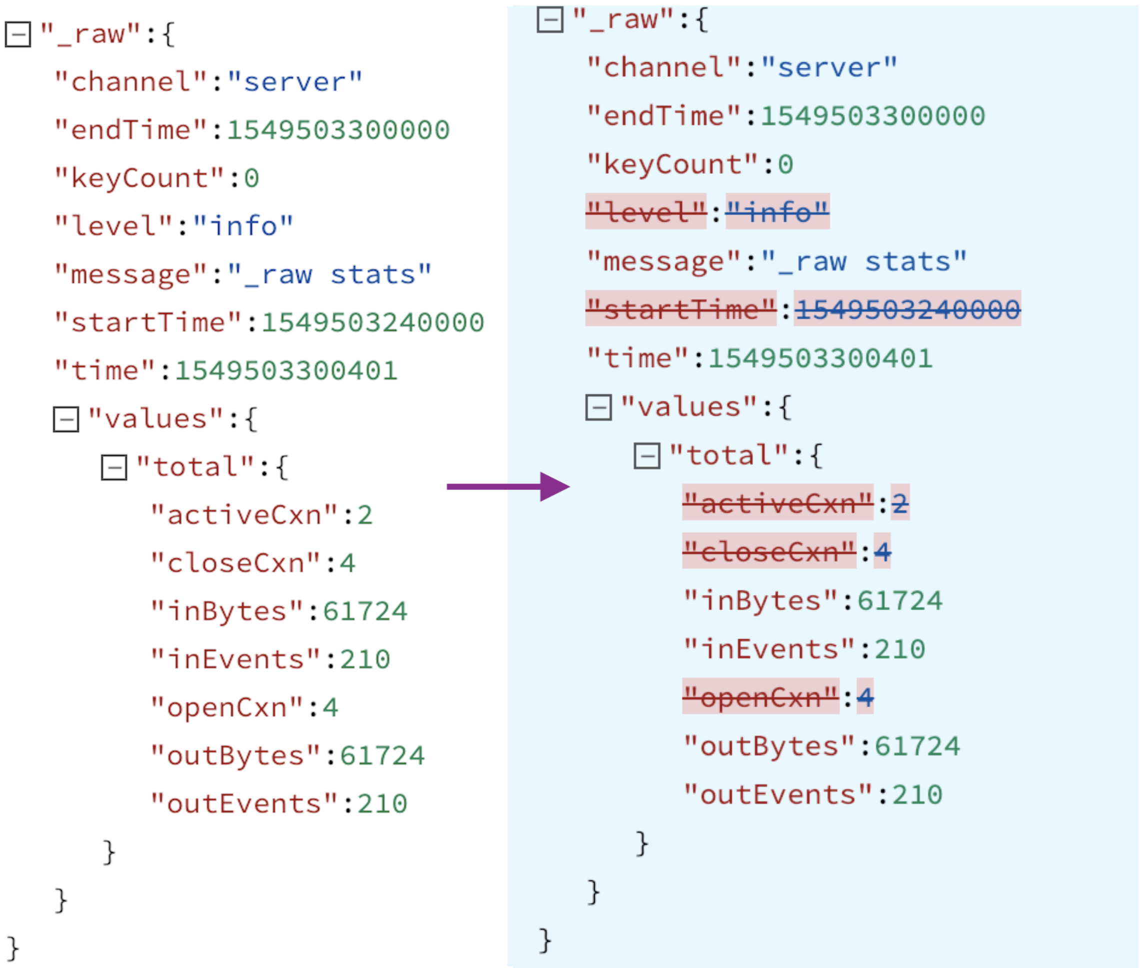
Fields To Remove: values.total.*Cxn, startTime

Fields Filter Expression: `!(name=='level' && value=='info')`

Destination Field:

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 **Preview** > **Add a Sample** > **Paste a Sample**:

```
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 **Parser 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: `Reserialize`
- 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: `Reserialize`
- Source field: [empty]
- Destination field: `bar`
- Type: `JSON`
- List of fields: [empty]
- Fields to keep: `a`, `b`, `c`, `d`

;

6.21. Publish Metrics

The Publish Metrics Function extracts, formats, and outputs metrics from events.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description about this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Overwrite: If set to `Yes`, overwrite previous metric specs. Otherwise, append. Defaults to `No`.

Metrics

Add Metrics: List of metrics to extract from the event and format. Destinations can pass the formatted metrics to a metrics aggregation platform. Click **Add Metrics** to add new rows containing the following options:

- **Event field name**: The name of the field (in the event) that contains the metric value. Should contain only letters, numbers, underscores (`_`), and `.` characters (to separate names in nested structures).
- **Metric name expression**: JavaScript expression to evaluate the metric field name. Defaults to the **Event field name** value.

  > The JavaScript expression will evaluate the metric field name only after the metrics are processed for transport to the Destination. While in the processing Pipeline, the metric name expression appears as a literal.

- **Metric type**: Select `Gauge` (the default), `Counter`, `Timer`, or `Distribution`. 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.)

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 `!_* *`.

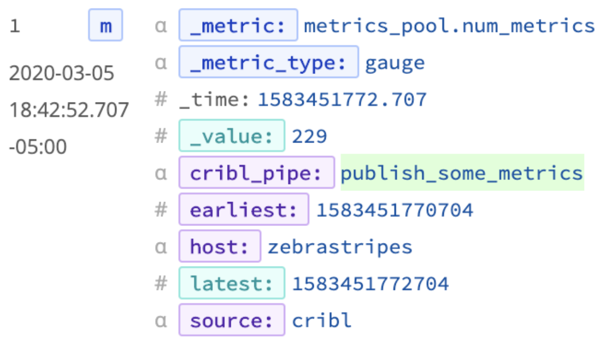
Remove Dimensions: Optional list of dimensions to associate with every extracted metric value. Leave blank if this function is used to process output from the Aggregation function as dimensions will be automatically discovered. If this Function is used to process output from the Aggregations Function, leave this field blank, because dimensions will be automatically discovered.

Overwrite: If set to `Yes`, overwrite previous metric specs. Otherwise, append. Defaults to `No`.

> The **Add Dimensions** and **Remove Dimensions** fields support wildcards and negated terms. When you use negated terms, the list is order-sensitive. E.g., `!foobar` before `foo*` means "All fields that start with `foo`, except `foobar`." However, `!foo*` before `*` means "All fields, except for those that start with `foo`."
>
> Cribl 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
2020-03-05               α _metric_type: gauge
18:42:52.707             # _time: 1583451772.707
-05:00                   # _value: 229
                         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**
packets	`metric_name.packets`	**Gauge**

Dimensions

```
action interface_id dstaddr
```

All specified dimension names must align with those from the original event. When you preview the Function's output, the metrics and dimensions will all have special highlighting to separate them from other fields. Additional highlighting is used to differentiate the metrics from the dimensions. (If one or more metrics/dimensions are not highlighted as expected, check the Function's configuration.)

Raw Output

```
metric_name.bytes:262256|g#action:REJECT,interface_id:eni-
02f03c2880e4aaa3,dstaddr:10.0.1.11
```

```
metric_name.packets:6556|g#action:REJECT,interface_id:eni-
02f03c2880e4aaa3,dstaddr:10.0.1.11
```

Compatible Destinations

All text after the # symbol represents the dimensions as key-value pairs. In order for dimension data to be included in metrics, the Destination type cannot be standard **StatsD**. However, **StatsD Extended**, **Splunk**, and **Graphite** do support dimensions.

Formatted Output

```
{
  "action": "REJECT",
  "interface_id": "eni-02f03c2880e4aaa3",
  "dstaddr": "10.0.1.11",
  "metric_name.bytes": 262256,
  "metric_name.packets": 6556,
}
```

Scenario B:

Assume that we want to extract some metrics from specific fields in PANOS logs, whose events have the following structure:

```
future_use_0,receive_time, serial_number, type, threat_content_type, future_use_1, generated_time, source_ip, destination_ip, nat_source_ip, nat_destination_ip, rule_name, source_user, destination_user, application, virtual_system, source_zone, destination_zone, inbound_interface, outbound_interface, log_action, future_use_2, session_id, repeat_count, source_port, destination_port, nat_source_port, nat_destination_port, flags, protocol, action, bytes, bytes_sent, bytes_received, packets, start_time, elapsed_time, category, future_use_3, sequence_number, action_flags, source_location, destination_location, future_use_4, packets_sent, packets_received, session_end_reason, device_group_hierarchy_level_1, device_group_hierarchy_level_2, device_group_hierarchy_level_3, device_group_hierarchy_level_4, virtual_system_name, device_name, action_source, source_vm_uuid, destination_vm_uuid, tunnel_id_imsi, monitor_tag_imei, parent_session_id, parent_start_time, tunnel_type, sctp_association_id, sctp_chunks, sctp_chunks_sent, sctp_chunks_received
```

For example:

```
Jan 10 10:19:15 DMZ-internal.nsa.gov 1,2019/01/10 10:19:15,001234567890002,TRAFFIC,drop,2304,2019/01/10 10:19:15,209.118.103.150,160.177.222.249,0.0.0.0,0.0.0.0,InternalServer,,,not-applicable,vsys1,inside,z1-FW-Transit,ethernet1/2,,All traffic,2019/01/10 10:19:15,0,1,63712,443,0,0,0x0,udp,deny,60,60,0,1,2019/01/10 10:19:15,0,any,0,0123456789,0x0,Netherlands,10.0.0.0-10.255.255.255,0,1,0,policy-deny,0,0,0,0,,DMZ-internal,from-policy,,,0,,0,,N/A,0,0,0,0,1202585d-b4d5-5b4c-aaa2-d80d77ba456e,0
```

Our goal is to use the four values of `bytes_sent`, `bytes_received`, `packets_sent`, and `packets_received` as metrics across these dimensions: `destination_ip`, `inbound_interface`, `outbound_interface`, and `destination_port`.

Here again, our Pipeline will need a Parser Function before the Publish Metrics Function.

Parser Function

Filter: Set as needed Operation mode: Extract Type: Extended Log File Format (automatically set when specifying a Library) Library: Palo Alto Traffic Source: `_raw` (No need to specify any other fields.)

Publish Metrics Function

Set up the Publish Metrics Function as follows.

Metrics

EVENT FIELD NAME	METRIC NAME EXPRESSION	METRIC TYPE
`bytes_sent`	`` `metric.${host}.bytes_sent` ``	Counter
`bytes_received`	`` `metric.${host}.bytes_rcvd` ``	Counter
`packets_sent`	`` `metric.${host}.pkts_sent` ``	Counter
`packets_received`	`` `metric.${host}.pkts_rcvd` ``	Counter

Added Dimensions

`destination_ip`, `inbound_interface`, `outbound_interface`, `destination_port`

Raw Output

```
metric.10.10.12.192.bytes_sent:60|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
metric.10.10.12.192.bytes_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
metric.10.10.12.192.pkts_sent:1|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
metric.10.10.12.192.pkts_rcvd:0|c|#destination_ip:160.177.222.249,inbound_interface:ethernet1/2,destination_port:443
```

Here again, all text after the `#` symbol represents the dimensions as key-value pairs. (See the Compatible Destinations note above.) Unlike the first example, this example uses JavaScript expressions, which you can see evaluated in the raw output where the `${host}` has been converted to `10.10.12.192`.

;

6.22. Redis

The Redis Function interacts with Redis stores, setting and getting key-hash and key-value combinations. Redis' in-memory caching of these key pairs enables large lookup tables that would be cumbersome with a .CSV or binary lookup file.

You can use Cribl Stream Collectors (e.g., a REST Collector) to retrieve reference data from desired endpoints, and then use this Function to store the data on Redis and retrieve it to enrich your production data. Note that Cribl Stream does not cache the data returned from this Redis 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`.

Result field: Name of the field in which to store the returned value. (Leave empty to discard the returned value.)

Command: Redis command to perform. Required. (A complete list of Redis commands is at: https://redis.io/commands.)

Key: A JavaScript expression to compute the value of the key to operate on. Can also be a constant, e.g.: `username`. This is a required field. Click the icon at right to open a validation modal.

Args: A JavaScript expression to compute arguments to the operation. Can return an array. Click the icon at right to open a validation modal.

Redis URL: Redis URL to connect to. The format is: `[redis[s]:]//[[user][:password@]][host][:port][/db-number][?db=db-number[&password=bar[&option=value]]]`

For example: `redis://user:secret@localhost:6379/0?foo=bar&qux=baz`

With no `user` specified: `redis://secret@localhost:6379/0?foo=bar&qux=baz`

Redis URL Vs. Redis ACL

> Through LogStream 2.4.3, the **Redis URL** field has limited compatibility with Redis 6.x's ACL (Access Control List) feature. When using an ACL, point this field 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 specific user other than `default`, or authentication against Redis will fail.

Authentication Method

Use the **Authentication method** buttons to select one of the following options, some of which display additonal 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.

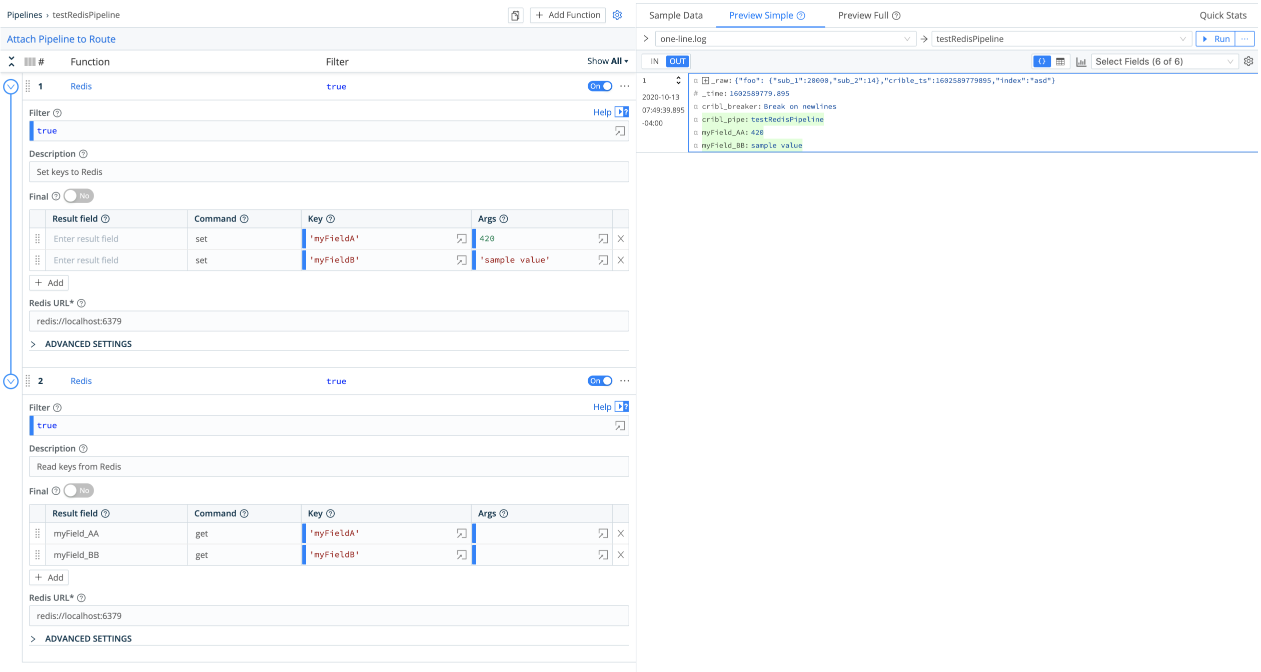
Advanced Settings

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.

Examples

Scenario A: Set and Get

This Pipeline demonstrates the use of a pair of Redis Functions. The first Function sets two key-value pairs in Redis. The second Function gets their values, by key, into two corresponding new **Result field**s.



#	Function	Filter	
1	Redis	true	On

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'

Redis URL*: redis://localhost:6379

#	Function	Filter	
2	Redis	true	On

Filter: `true`

Description: Read keys from Redis

Final: No

Result field	Command	Key	Args
myField_AA	get	'myFieldA'	
myField_BB	get	'myFieldB'	

Redis URL*: redis://localhost:6379

one-line.log → testRedisPipeline

Select Fields (6 of 6)

```
1          _raw: {"foo": {"sub_1":20000,"sub_2":14},"crible_ts":1602589779895,"index":"asd"}
2020-10-13 _time: 1602589779.895
07:49:39.895 cribl_breaker: Break on newlines
-04:00     cribl_pipe: testRedisPipeline
           myField_AA: 420
           myField_BB: sample value
```

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

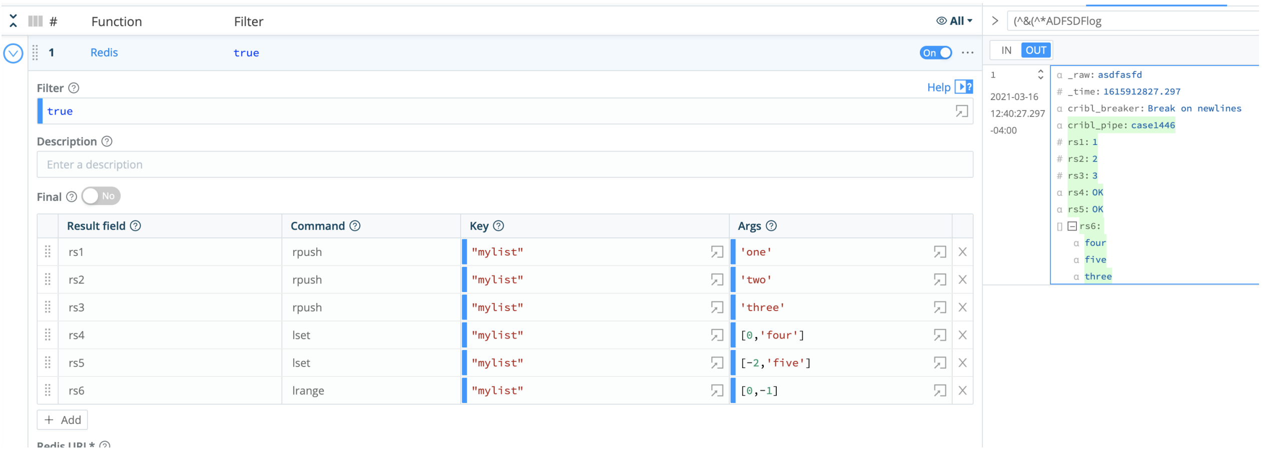
Description: `Read keys from Redis`

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`).



1 Redis true

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
α _raw: asdfasfd
# _time: 1615912827.297
α cribl_breaker: Break on newlines
α cribl_pipe: case1446
# rs1: 1
# rs2: 2
# rs3: 3
α rs4: OK
α rs5: OK
[] rs6:
    α four
    α five
    α 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"
      }
    ]
  }
}
```

;

6.23. Regex Extract

The Regex Extract Function extracts fields using regex named groups. (In Splunk, these will be index-time fields). Fields that start with `__` (double underscore) are special in Cribl 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

Example 1: Single Field from Simple Event

Assume a simple event that looks like this: `metric1=23 metric2=42 dc=23 abc=xyz`

Extract **only** the `metric1` field:

Regex: `metric1=(?<metric1>\d+)` **Result**: `metric1:"23"`

Example 2: Key-Value Pairs from Multiple Fields

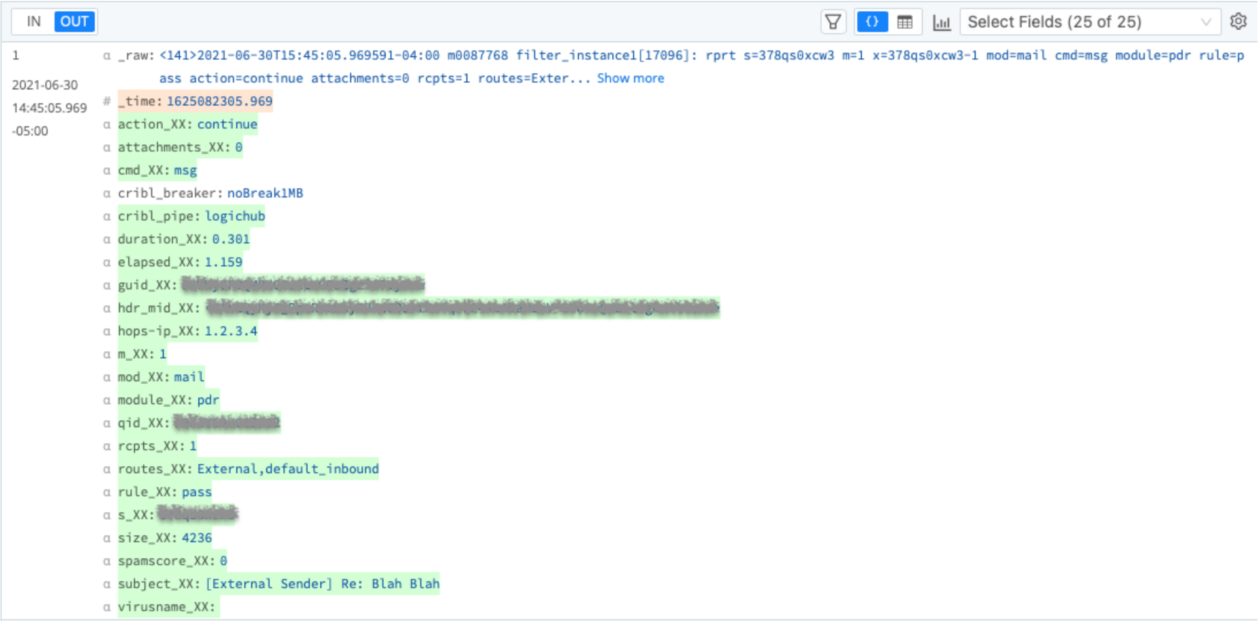
Use this sample:

```
rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src="00000000-0000-0000-0000-000000000000" ssl_server_cert_status="Not Checked" dest_ip=172.20.115.42 sec_intel_event=No mac_address=00:00:00:00:00:00 dest_bytes=3746 dest_autonomous_system=0 security_context=00000000000000000000000000000000 src_port=41925 web_app=Unknown url=https://outlook.ssg.petsmart.com url_reputation="Risk unknown" first_pkt_sec=1543598207 vlan_id=0 ssl_flow_error=0 ssl_actual_action=Unknown has_ipv6=1 monitor_rule_6=N/A monitor_rule_7=N/A monitor_rule_4=N/A monitor_rule_5=N/A monitor_rule_2=N/A monitor_rule_3=N/A ips_count=0 monitor_rule_1=N/A dest_tos=0 src_ip=192.168.228.5 referenced_host="" iface_ingress=DMZ3.30 monitor_rule_8=0 event_subtype=1 fw_rule_reason=N/A event_type=1003 ssl_version=Unknown dns_resp_id=0 sensor=ssg-inet-fpr-ftd-fw01 sec_zone_egress=Inside src_tos=0 client_app="SSL client" snmp_in=0 user=Unknown ssl_flow_messages=0 iface_egress=inside http_referrer="" src_pkts=0 event_desc="Flow Statistics" event_usec=0 client_version="" fw_rule_action=Allow ssl_cert_fingerprint=0000000000000000000000000000000000000000 ssl_url_category=0 file_count=0 sec_zone_ingress=DMZ3 instance_id=6 src_bytes=1013 src_ip_country=unknown ssl_cipher_suite=TLS_NULL_WITH_NULL_NULL user_agent="" http_response=0 src_mask=0 dest_mask=0 sec_intel_ip=N/A netbios_domain="" tcp_flags=0 dns_rec_id=0 fw_policy="SSG INET Access Control Policy" last_pkt_sec=1543598207 legacy_ip_address=0.0.0.0 ip_proto=TCP connection_id=21378 dest_pkts=0 app_proto=HTTPS ssl_flow_status=Unknown ssl_rule_id=0 ssl_session_id=0000000000000000000000000000000000000000000000000000000000000000 dns_query="" rec_type_desc="Connection Statistics" url_category=Unknown fw_rule="Outbound Web" src_autonomous_system=0 ssl_flow_flags=0 ip_layer=0 event_sec=1543598205 ssl_ticket_id=0000000000000000000000000000000000000000 sinkhole_uuid=00000000-0000-0000-0000-000000000000 dest_ip_country=unknown ssl_expected_action=Unknown num_ioc=0 dns_ttl=0 ssl_policy_id=00000000000000000000000000000000 ssl_server_name=""
```

Use a regex to extract **all** k=v pairs, then use **Field Name Format Expression** to append an `_XX` suffix to each extracted field:

Regex: `(?<_NAME_0>[\w-]+)="?(?<_VALUE_0>(?<=")[^"]*|\S*)` **Field Name Format Expression**: `${name}_XX`

Results:



```
1                   α _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
                          ass action=continue attachments=0 rcpts=1 routes=Exter... Show more
2021-06-30          # _time: 1625082305.969
14:45:05.969        α action_XX: continue
-05:00              α attachments_XX: 0
                    α cmd_XX: msg
                    α cribl_breaker: noBreak1MB
                    α cribl_pipe: logichub
                    α duration_XX: 0.301
                    α elapsed_XX: 1.159
                    α guid_XX: [redacted]
                    α hdr_mid_XX: [redacted]
                    α hops-ip_XX: 1.2.3.4
                    α m_XX: 1
                    α mod_XX: mail
                    α module_XX: pdr
                    α qid_XX: [redacted]
                    α rcpts_XX: 1
                    α routes_XX: External,default_inbound
                    α rule_XX: pass
                    α s_XX: [redacted]
                    α size_XX: 4236
                    α spamscore_XX: 0
                    α subject_XX: [External Sender] Re: Blah Blah
                    α virusname_XX:
```

Example 2 results

Example 3: Multi-Stage Extraction, Complex Events

This example builds on the syntax in Example 2, to tackle a more complex event structure.

In the right **Sample Data** pane, click **Paste** and insert the following sample:

```
<134>1 2020-12-22T17:06:08Z CORP_INT_NLB CheckPoint 18160 - [action:"Accept";
conn_direction:"Internal"; flags:"4606212"; ifdir:"inbound"; ifname:"bond2.1025";
logid:"0"; loguid:"{0x5fe25889,0x0,0x80ad57cd,0xeb91c0c3}"; origin:"192.168.20.54";
originsicname:"CN=TST32-VSX0-FW-DC-01_tst302-shd,O=CORP-SEC-SHRD-CMA..t7xpcz";
sequencenum:"3"; time:"1608656768"; version:"5"; __policy_id_tag:"product=VPN-1 &
FireWall-1[db_tag={15E4B45A-663B-5B49-BD59-
CD9B9F21AA16};mgmt=SHRDFW01CON;date=1608236862;policy_name=TEST-SHRD-POL\]";
dst:"192.168.79.20"; log_delay:"1608656768"; layer_name:”TEST-SHRD-POL Security";
layer_uuid:"e914c2f3-d7bd-4a77-8e7a-7a5e403447aa"; match_id:"1"; parent_rule:"0";
rule_action:"Accept"; rule_uid:"001ab86d-d201-4b61-9b64-0fede1a9f059"; product:"VPN-
1 & FireWall-1"; proto:"17"; s_port:"45519"; service:"123"; service_id:"ntp-udp";
src:"192.168.79.22"; ]
```

This event is from a CheckPoint Firewall CMA system. With this type of event structure, properly extracting each event field into a separate metadata field requires two-stage processing. So we'll use two Regex Extract Functions.

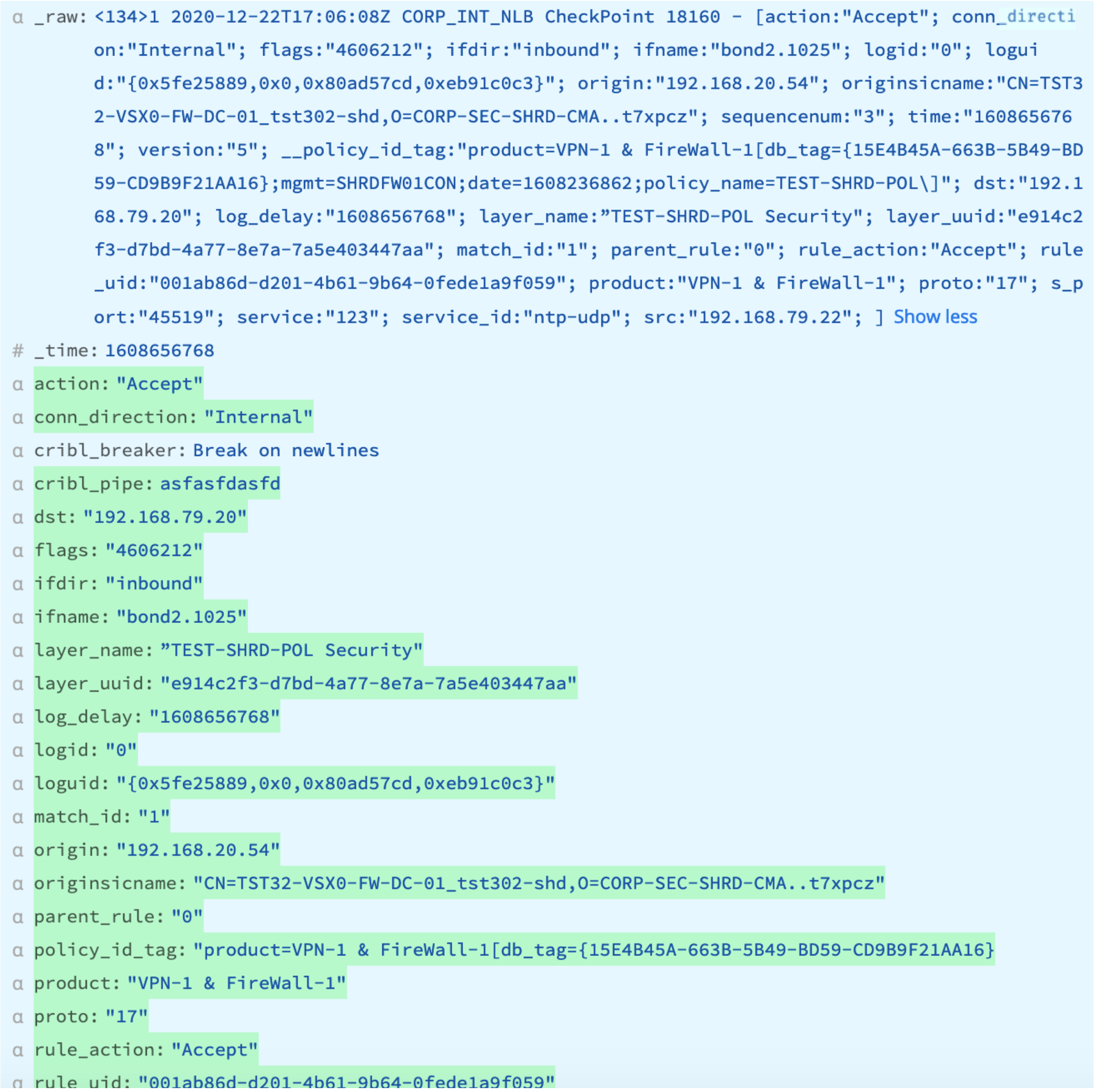
The first Regex Function splits the event to separate the actual data from the header information. We'll split after the `CheckPoint 18160` string, by capturing everything between the `[` and `]`:

Regex: `\[(?<__fields>.*)\]` **Source**: `_raw`

Next, add this second Regex Extract Function to extract all k=v pairs:

Regex: `(?<_NAME_0>[^ :]+):(?<_VALUE_0>[^;]+);` **Source**: `__fields`

Results:



```
a _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
a action: "Accept"
a conn_direction: "Internal"
a cribl_breaker: Break on newlines
a cribl_pipe: asfasfdasfd
a dst: "192.168.79.20"
a flags: "4606212"
a ifdir: "inbound"
a ifname: "bond2.1025"
a layer_name: ”TEST-SHRD-POL Security"
a layer_uuid: "e914c2f3-d7bd-4a77-8e7a-7a5e403447aa"
a log_delay: "1608656768"
a logid: "0"
a loguid: "{0x5fe25889,0x0,0x80ad57cd,0xeb91c0c3}"
a match_id: "1"
a origin: "192.168.20.54"
a originsicname: "CN=TST32-VSX0-FW-DC-01_tst302-shd,O=CORP-SEC-SHRD-CMA..t7xpcz"
a parent_rule: "0"
a policy_id_tag: "product=VPN-1 & FireWall-1[db_tag={15E4B45A-663B-5B49-BD59-CD9B9F21AA16}
a product: "VPN-1 & FireWall-1"
a proto: "17"
a rule_action: "Accept"
a rule_uid: "001ab86d-d201-4b61-9b64-0fede1a9f059"
```

Example 3 results

> For further examples, see Using Cribl to Analyze DNS Logs in Real Time – Part 2.

;

6.24. Regex Filter

The Regex Filter Function filters out events based on regex matches.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Regex: Regex to test against. Defaults to empty.

Additional regex: Click **+ Add Regex** to chain extra regex conditions.

Field: Name of the field to test against the regex. Defaults to `_raw`. Supports nested addressing.

Examples

See Regex Filtering for examples.

;

6.25. Rename

The Rename Function is designed to change fields' names or reformat their names (e.g., by normalizing names to camelcase). You can use Rename to change specified fields (much like the Eval Function), or for bulk renaming based on a JavaScript expression (much like the Parser Function).

Compared to these alternatives, Rename offers a streamlined way to alter only field names, without other effects.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Optionally, enter a simple description of this step in the Pipeline. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Parent fields: Specify fields whose children will inherit the **Rename fields** and **Rename expression** operations. Supports wildcards. If empty, only top-level fields will be renamed.

> This Function cannot operate on internal fields whose names begin with double underscores (`__`).
>
> Note that you can still use `__e` to access fields within the `(context)` object, as long as those fields do not begin with double underscores. That's because `__e` is a special variable, not a field.

Rename fields: Each row here is a key-value pair that defines how to rename fields. The current name is the key, and the new name is the value. Click **+ Add Field** to add more rows.

- **Current name**: Original name of the field to rename. You must quote literal identifiers (non-alphanumeric characters such as spaces or hyphens).

- **New name**: New or reformatted name for the field. Here again, you must quote literals.

Rename expression: Optional JavaScript expression whose returned value will be used to rename fields. Use the `name` and `value` global variables to access fields' names/values. Example: `name.startsWith('data') ? name.toUpperCase() : name`. You can access other fields' values via `event.<fieldName>`.

> A single Function can include both **Rename fields** (to rename specified field names) and **Rename expression** (to globally rename fields). However, the **Rename fields** strategy will execute first.

Advanced Settings

Parent field wildcard depth: For wildcards specified in **Parent fields**, sets the maximum depth within events to match and rename fields. Enter `0` to match only top-level fields. Defaults to `5` levels down.

Example

Change the `level` field, and all fields that start with `out`, to all-uppercase.

Example event:

```
{"inEvents": 622,
  "level": "info",
  "outEvents": 311,
  "outBytes": 144030,
  "activeCxn": 0,
  "openCxn": 0,
  "closeCxn": 0,
  "activeEP": 105,
  "blockedEP": 0
}
```

Rename Fields:

Current name: `level`
New name: `LEVEL` **Rename expression**: `name.startsWith('out') ? name.toUpperCase() : name`

Event after Rename:

```
{"inEvents": 622,
  "LEVEL": "info",
  "OUTEVENTS": 311,
  "OUTBYTES": 144030,
  "activeCxn": 0,
  "openCxn": 0,
  "closeCxn": 0,
  "activeEP": 105,
  "blockedEP": 0
}
```

Applications

1. Remove filename prefix `<myPrefix>`:

   Rename expression: `name.replace(/<myPrefix>/, '')`

2. Add a wildcard to rename a set of fields named `json.record[0]`, `json.record[1]`, etc., preserving the variable numbers in the brackets:

   Rename expression: `name.replace(/(json)\.(\w+)/,'MYNEWNAME-$2-$1')`

;

6.26. Rollup Metrics

The Rollup Metrics Function merges/rolls up frequently generated incoming metrics into more manageable time windows.

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`.

;

6.27. Sampling

The Sampling Function filters out events, based on an expression and a sampling rate.

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.

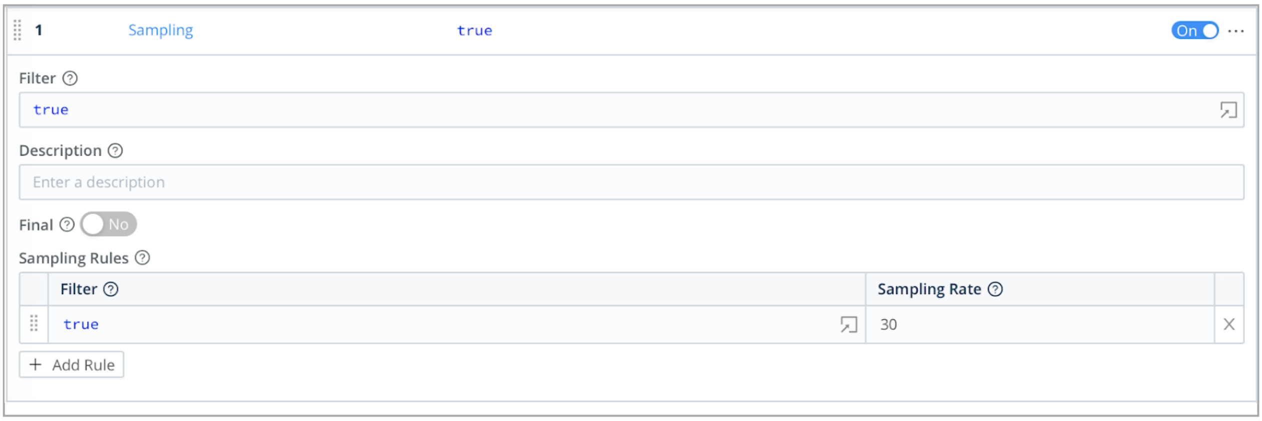
Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

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.



1 Sampling `true` On

Filter: `true`

Description: Enter a description

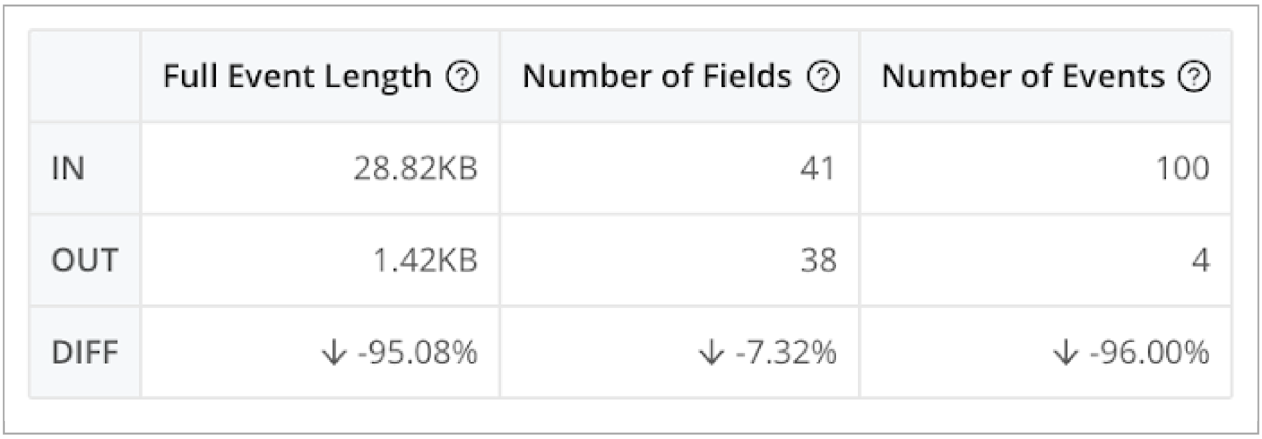
Final: No

Sampling Rules

Filter	Sampling Rate
`true`	30

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.

> Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

;

6.28. Serialize

Use the Serialize Function to serialize an event's content into a predefined format.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Type: Data output format. Defaults to `CSV`.

Library: Browse Parser/Formatter library.

Fields to serialize: Required for `CSV`, `ELFF`, and `CLF` Types. (All other formats support wildcard field lists.)

Source field: Field containing the object to serialize. Leave blank to serialize top-level event fields.

Destination field: Field to serialize the data into. Defaults to `_raw`.

Examples

Scenario A: JSON to CSV

Assume a simple event that looks like this: `{"time":"2019-08-25T14:19:10.240Z","channel":"input","level":"info","message":"initializing input","type":"kafka"}`

We want to serialize these fields: `_time`, `channel`, `level`, and `type` into a single string, in CSV format, stored in a new destination field called `test`.

To properly extract the key-value pairs from this event structure, we'll use a built-in Event Breaker:

1. Copy the above sample event to your clipboard.
2. In the **Preview** pane, select **Paste a Sample**, and paste in the sample event.

3. Under **Select Event Breaker**, choose **ndjson** (newline-delimited JSON), and click **Save as a Sample File**.

Now you're ready to configure the Serialize Function, using the settings below:

Type: `CSV` Fields to Serialize: `_time channel level type` Destination Field: `test` Source Field: [leave empty] **Result**: `test: 1566742750.24,input,info,kafka`

In the new `test` field, you now see the `time`, `channel`, `level`, and `type` keys extracted as top-level fields.

Scenario B: CSV to JSON

Let's assume that a merchant wants to extract a subset of each customer order, to aggregate anonymized order statistics across their customer base. The transaction data is originally in CSV format, but the statistical data must be in JSON.

Here's a CSV header (which we don't want to process), followed by a row that represents one order:

`orderID,custName,street,city,state,zip 20200622102822,john smith,100 Main St.,Anytown,AK,99911`

To convert to JSON, we'll need to first parse each field from the CSV to a manipulable field in the Pipeline, which the Serialize Function will be able to reference. In this example, the new manipulable field is `message`.

Use the `Parser` Function:

Filter: `true` Operation mode: `Extract` Type: `CSV` Source field: `_raw` Destination field: `message` List of fields: `orderID custName street city state zip`

Now use the Serialize Function:

Filter: `true` Type: `JSON` Fields to serialize: `city state` Source field: `message` Destination field: `orderStats`

;

6.29. Suppress

The Suppress Function suppresses events over a time period, based on evaluating a key expression.

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

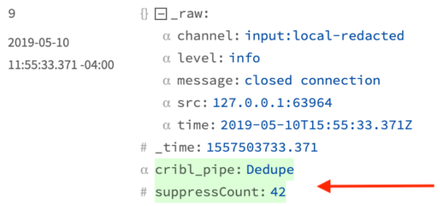
Maximum cache size : The maximum number of keys that can be cached before idle entries are removed. Before changing the default `50000`, contact Cribl Support to understand the implications.

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!=undefined && port!=undefined` 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.
>
> Each Worker Process executes this Function independently on its share of events. For details, see Functions and Shared-Nothing Architecture.

;

6.30. Tee

The Tee Function tees events out to a command of choice, via `stdin`. The output is one JSON-formatted event per line. You can send the events to (for example) a local file on the Cribl 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.

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_r
eceived":"0"}
```

> In this Function's **Command** field, you can specify commands other than `tee` itself. For example: By using `nc` as the command, and specifying `localhost` and a port number (as two separate arguments), you'll see event data being received via `nc` on the specified port.

;

6.31. Trim Timestamp

The Trim Timestamp Function removes timestamp patterns from events, and (optionally) stores them in a specified field.

This Function looks for a timestamp pattern that exists between the characters indicated by numeric `timestartpos` and `timeendpos` fields. It removes `timestartpos` and `timeendpos` along with the timestamp pattern.

> The Trim Timestamp Function, in current Cribl 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

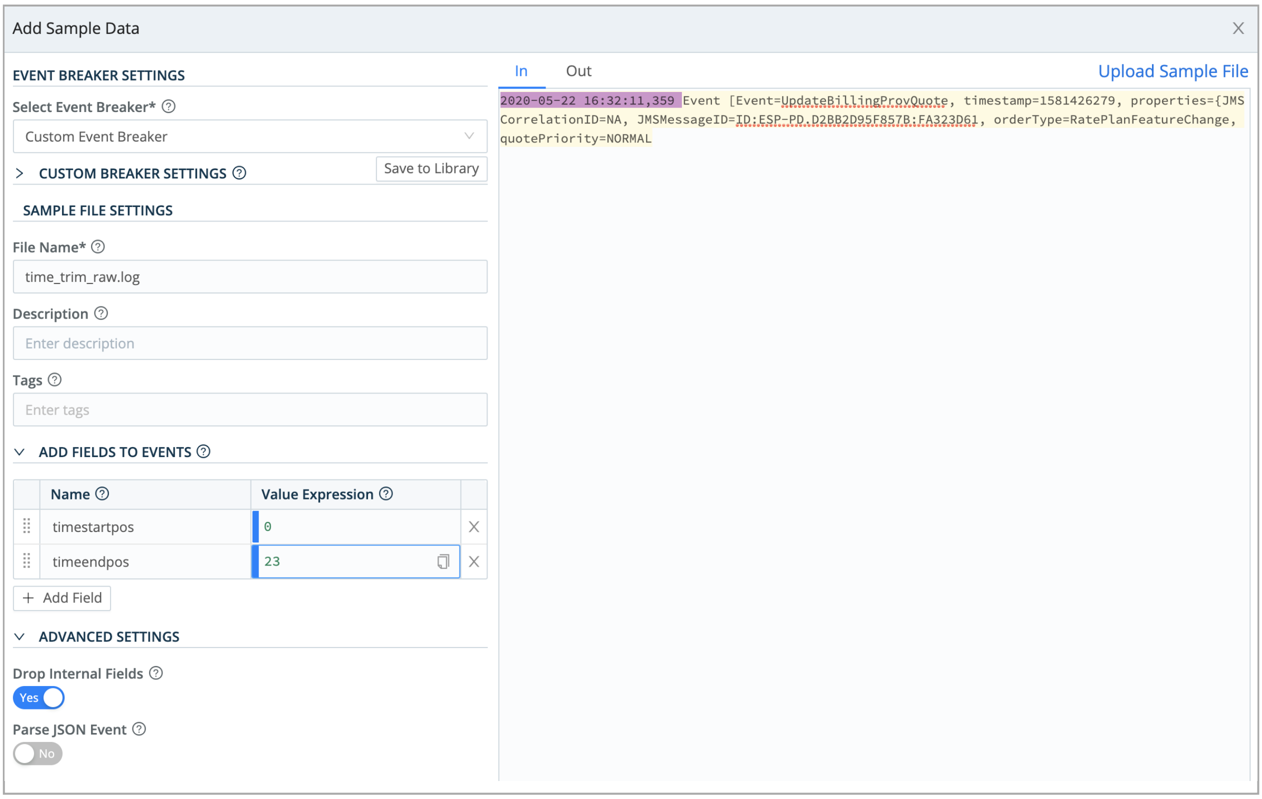
Remove the timestamp pattern (indicated by `timestartpos` and `timeendpos`) from `_raw`, and stash it in a field called `time_field`.

Field name: `time_field`

Example event before:

```
{"_raw": "2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote,
timestamp=1581426279, properties={JMSCorrelationID=NA, JMSMessageID=ID:ESP-
PD.D2BB2D95F857B:FA323D61, orderType=RatePlanFeatureChange, quotePriority=NORMAL}",
"timestartpos":0,
"timeendpos":23
}
```

To create this example payload, we selected **Sample Data** > **Paste**, pasted the `_raw` field's original contents into the resulting modal, and then added the two required position fields:



Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: Custom Event Breaker

CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*: time_trim_raw.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

Name	Value Expression
timestartpos	0
timeendpos	23

ADVANCED SETTINGS

Drop Internal Fields: Yes

Parse JSON Event: No

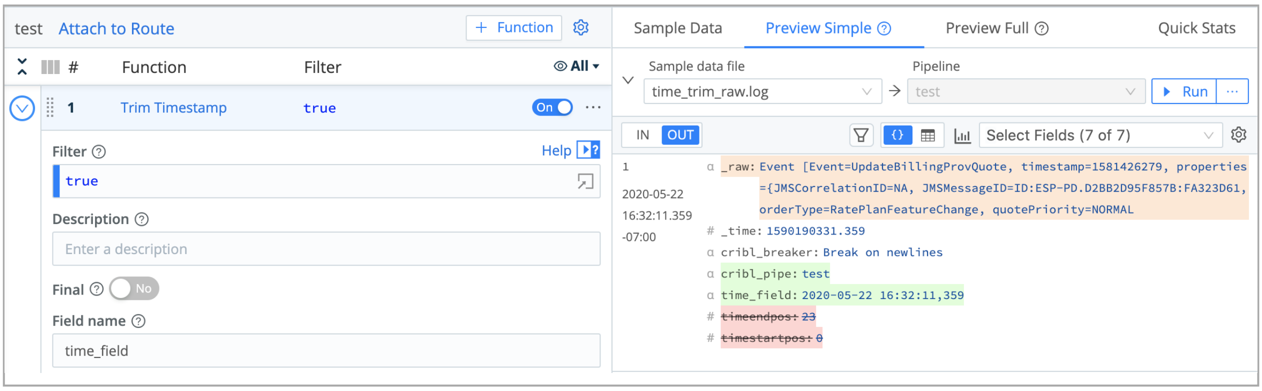
```
2020-05-22 16:32:11,359 Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties={JMS
CorrelationID=NA, JMSMessageID=ID:ESP-PD.D2BB2D95F857B:FA323D61, orderType=RatePlanFeatureChange,
quotePriority=NORMAL
```

Example event setup

Example event after:

```
{"_raw": "Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties=
{JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.D2BB2D95F857B:FA323D61,
orderType=RatePlanFeatureChange, quotePriority=NORMAL}",
"time_field":"2020-05-22 16:32:11,359"
}
```

In the Preview pane's **OUT** view, the original timestamp has been removed from `_raw`, and lifted into the new `time_field` we specified in the Function. The `timestartpos` and `timeendpos` fields have been removed.



1 Trim Timestamp true On

Filter: true

Description: Enter a description

Final: No

Field name: time_field

Sample data file: time_trim_raw.log → Pipeline: test

Select Fields (7 of 7)

```
1                   α _raw: Event [Event=UpdateBillingProvQuote, timestamp=1581426279, properties
                            ={JMSCorrelationID=NA, JMSMessageID=ID:ESP-PD.D2BB2D95F857B:FA323D61,
2020-05-22                  orderType=RatePlanFeatureChange, quotePriority=NORMAL
16:32:11.359        # _time: 1590190331.359
-07:00              α cribl_breaker: Break on newlines
                    α cribl_pipe: test
                    α time_field: 2020-05-22 16:32:11,359
                    # ~~timeendpos: 23~~
                    # ~~timestartpos: 0~~
```

Example event, saved and transformed

;

6.32. Unroll

The Unroll Function accepts an array field – or an expression to evaluate an array field – and breaks/unrolls the array into individual events.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Source field expression: Field in which to find/calculate the array to unroll. E.g.: `_raw`, `_raw.split(/\n/)`. Defaults to `_raw`.

Destination field: Field (within the destination event) in which to place the unrolled value. Defaults to `_raw`.

Example

Assume we want to break/unroll each line of this event:

```
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02
/lib/systemd/systemd --system --deserialize 28
root         2  0.0  0.0      0     0 ?        S     2018   0:00 [kthreadd]
root         3  0.0  0.0      0     0 ?        S     2018   1:51 [ksoftirqd/0]
root         5  0.0  0.0      0     0 ?        S<    2018   0:00 [kworker/0:0H]
root         7  0.0  0.0      0     0 ?        S     2018   3:55 [rcu_sched]
root         8  0.0  0.0      0     0 ?        S     2018   0:00 [rcu_bh]
```

Settings

Source field expression: `_raw.split(/\n/)`

> The `split()` JavaScript method breaks `_raw` into an ordered set of substrings/values, puts these values into an array, and returns the array.

Destination field: `_raw`

```
Event 1:
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND

Event 2:
root         1  0.0  0.5  38000  5356 ?        Ss    2018   2:02
/lib/systemd/systemd --system --deserialize 28

Event 3:
root         2  0.0  0.0      0     0 ?        S     2018   0:00 [kthreadd]

Event 4:
root         3  0.0  0.0      0     0 ?        S     2018   1:51 [ksoftirqd/0]

Event 5:
root         5  0.0  0.0      0     0 ?        S<    2018   0:00 [kworker/0:0H]

Event 6:
root         7  0.0  0.0      0     0 ?        S     2018   3:55 [rcu_sched]

Event 7:
root         8  0.0  0.0      0     0 ?        S     2018   0:00 [rcu_bh]
```

;

6.33. XML Unroll

The XML Unroll Function accepts a proper XML event with a set of elements, and converts the elements into individual events.

Usage

Filter: Filter expression (JS) that selects data to feed through the Function. Defaults to `true`, meaning it evaluates all events.

Description: Simple description of this Function. Defaults to empty.

Final: If toggled to `Yes`, stops feeding data to the downstream Functions. Defaults to `No`.

Unroll elements regex: Path to the array to unroll. E.g.: `^root\.child\.ElementToUnroll$`

Copy elements regex: Regex matching elements to copy into each unrolled event. E.g.: `^root\.(childA|childB|childC)$`

Unroll index field: Cribl 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 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>
```

;

6.34. Prometheus Publisher (Deprecated)

> This Function is deprecated as of Cribl Stream 3.0. 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="ethernet1/2",destination_port="443
60

# TYPE perf_192_168_1_248_bytes_rcvd counter
metric_192_168_1_248_bytes_rcvd
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
0

# TYPE perf_192_168_1_248_pkts_sent counter
metric_192_168_1_248_pkts_sent
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
1

# TYPE perf_192_168_1_248_pkts_rcvd counter
metric_192_168_1_248_pkts_rcvd
{destination_ip="160.177.222.249",inbound_interface="ethernet1/2",destination_port="443
0
```

Now, we need to have Prometheus scrape the metrics. In this very basic example, you can add the target endpoint to the `prometheus.yml` file, under the `scrape_configs` -> `static_configs` section. Specify the endpoint in `IP:port` syntax, because Prometheus assumes (and requires) `/metrics` for all endpoints.

Restart Prometheus. Within just a few seconds, you should be able to use its query interface to retrieve metrics published by Cribl Stream.

;

6.35. Reverse DNS (deprecated)

The Reverse DNS Function resolves hostnames from a numeric IP address, using a reverse DNS lookup.

> This Function is deprecated. Use the DNS Lookup Function's reverse lookup feature instead.

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.

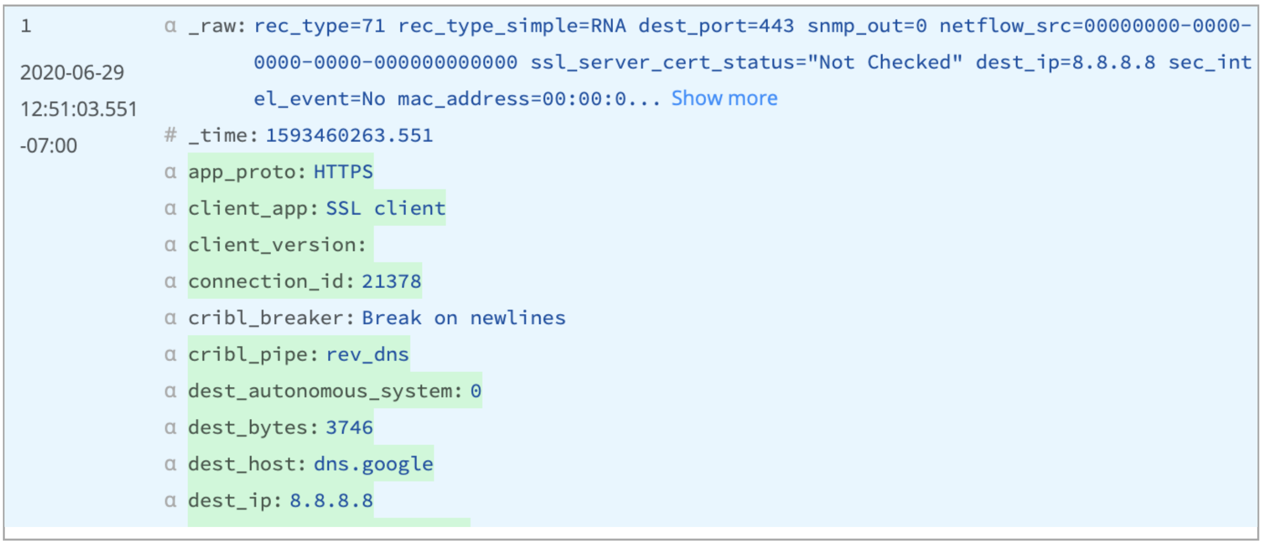
> If the field value is not in IPv4 or IPv6 format, the lookup is skipped.

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.

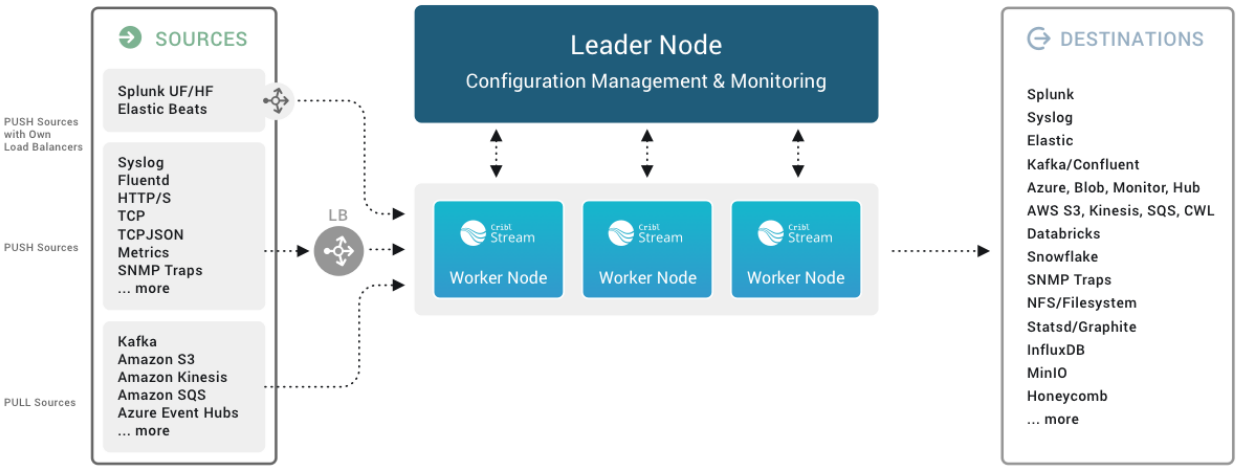


```
1            α _raw: rec_type=71 rec_type_simple=RNA dest_port=443 snmp_out=0 netflow_src=00000000-0000-
                     0000-0000-000000000000 ssl_server_cert_status="Not Checked" dest_ip=8.8.8.8 sec_int
2020-06-29           el_event=No mac_address=00:00:0... Show more
12:51:03.551 # _time: 1593460263.551
-07:00       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
```

;

7. Sources

Cribl Stream can receive continuous data input from various Sources, including Splunk, HTTP, Elastic Beats, Kinesis, Kafka, TCP JSON, and many others.



Push and Pull Sources

COLLECTOR Sources

Collectors, the top group of Sources in Cribl Stream's UI, are designed to ingest data **intermittently** – in on-demand bursts ("ad hoc collection"), or on preset schedules, or by "replaying" data from local or remote stores:

- Azure Blob Storage
- Filesystem/NFS
- Google Cloud Storage
- REST/API Endpoint
- S3
- Script
- Splunk Search

For background and instructions on using Collectors, see:

- Collectors
- Scheduling and Running
- Job Limits

PUSH Sources

Supported data Sources that **send** to Cribl Stream:

- Syslog
- TCP JSON
- Splunk TCP
- Splunk HEC
- Amazon Kinesis Firehose
- Prometheus Remote Write
- HTTP/S (Bulk API)
- Raw HTTP/S
- Elasticsearch API
- Metrics
- SNMP Trap
- TCP (Raw)
- Datadog Agent
- OpenTelemetry (OTel)
- AppScope
- Grafana
- Loki
- Windows Event Forwarder

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:

- Amazon Kinesis Streams
- Amazon SQS
- Amazon S3
- Google Cloud Pub/Sub
- Azure Event Hubs

- Azure Blob Storage
- Office 365 Services
- Office 365 Activity
- Office 365 Message Trace
- CrowdStrike
- Prometheus Scraper
- Kafka
- Confluent Cloud
- Splunk Search

System and Internal Sources

Sources that supply information generated by Cribl Stream about itself or from files it owns; or, that move data between Workers within your Cribl Stream deployment.

- Datagen
- Cribl Internal
- System Metrics
- Cribl HTTP
- Cribl TCP
- Cribl Stream (Deprecated)
- File Monitor
- Exec

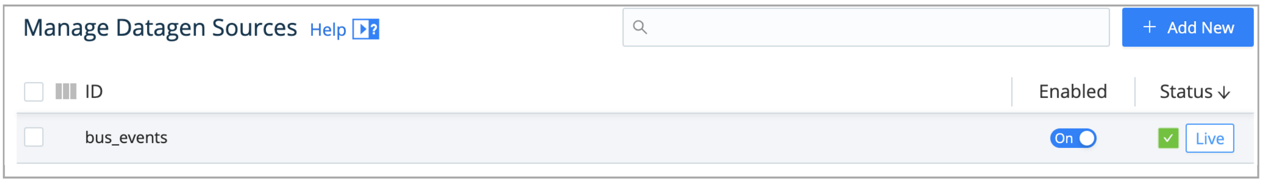
Configuring and Managing Sources

For each Source *type*, you can create multiple definitions, depending on your requirements.

To configure Sources, select **Data** > **Sources** from Cribl Stream's global top nav (single-instance deployments), or from a Worker Group's/Fleet's top nav (distributed deployments). On the resulting **Data Sources** page's tiles or left menu, select the desired type, then click **+ Add New**.

Capturing Source Data

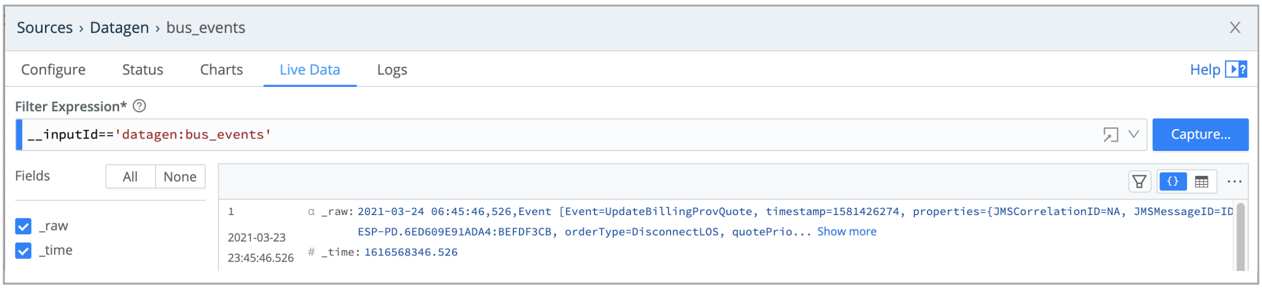
To capture data from a single enabled Source, you can bypass the Preview pane, and instead capture directly from a **Manage Sources** page. Just click the **Live** button beside the Source you want to capture.



□	ID	Enabled	Status
□	bus_events	On	✓ Live

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.



Filter Expression*

```
__inputId=='datagen:bus_events'
```

☑ _raw
☑ _time

```
1               α _raw: 2021-03-24 06:45:46,526,Event [Event=UpdateBillingProvQuote, timestamp=1581426274, properties={JMSCorrelationID=NA, JMSMessageID=ID
                       ESP-PD.6ED609E91ADA4:BEFDF3CB, orderType=DisconnectLOS, quotePrio... Show more
2021-03-23
23:45:46.526    # _time: 1616568346.526
```

Source modal > Live Data tab

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
- **Cribl Internal** > **CriblLogs** – Internal
- **Cribl Internal** > **CriblMetrics** – Internal

Backpressure Behavior and Persistent Queues

On the Destination side, you can configure how each Cribl Stream output will respond to a **backpressure** situation – a situation where its in-memory queue is overwhelmed with data.

All Destinations default to **Block** mode, in which they will refuse to accept new data until the downstream receiver is ready. Here, Cribl Stream will back-propagate block signals through the Source, all the way back to the sender (if it supports backpressure, too).

All Destinations also support **Drop** mode, which will simply discard new events until the receiver is ready.

Persistent Queues

Several Destinations also support a **Persistent Queue** option to minimize data loss. Here, the Destination will write data to disk until the receiver is ready. Then it will drain the disk-buffered data in FIFO (first in, first out) order. You can also define fallback behavior when the queue's allotted disk space is full.

Push Sources' config modals provide a corresponding **Persistent Queue Settings** option for inbound streaming data. When you enable this, 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 all the above modes and options, 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.

;

7.1. 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.
> - 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.
- Filesystem/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.
- 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 Filesystem/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 **Collectors** from the top nav. 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 Filesystem, 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** (side or top nav) > **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.

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

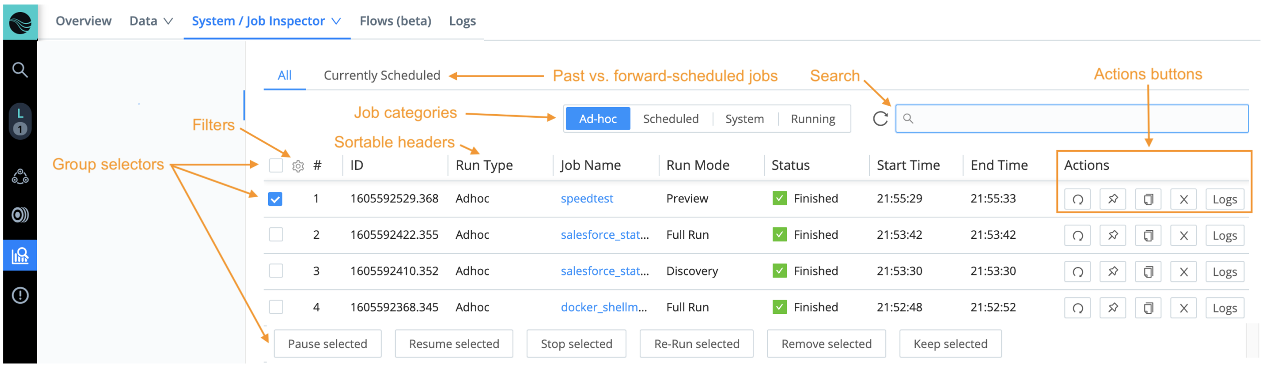
In a distributed deployment, you configure Collectors at the Worker Group level, and Worker Nodes execute the tasks. However, the Leader Node oversees the task distribution, and tries to maintain a fair balance across jobs.

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 global ⚙ **Settings** (lower left) > **General Settings > Job Limits**, 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 you configure Collectors' resource allocation globally in the Leader's global ⚙ **Settings** (lower left) > **General Settings > Job Limits** section.
>
> 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** (side or top nav) > **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
☑ 1	1605592529.368	Adhoc	speedtest	Preview	✓ Finished	21:55:29	21:55:33	Logs
□ 2	1605592422.355	Adhoc	salesforce_stat...	Full Run	✓ Finished	21:53:42	21:53:42	Logs
□ 3	1605592410.352	Adhoc	salesforce_stat...	Discovery	✓ Finished	21:53:30	21:53:30	Logs
□ 4	1605592368.345	Adhoc	docker_shellm...	Full Run	✓ Finished	21:52:48	21:52:52	Logs

Job Inspector: all the things

Here are the options available on the Job Inspector page:

- **All** vs. **Currently Scheduled** tabs: Click **Currently Scheduled** to see jobs foward-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.)

- Filters: Click the gear icon to open a drop-down with multiple options to filter the jobs shown within your selected category.

- 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: **Re-run**; **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).

Last updated by: Dritan Bitincka

;

7.1.1. Azure Blob Storage

Cribl Stream supports collecting data, and replaying specific events, from Azure Blob Storage. This page covers how to configure the Collector.

Configuring an Azure Blob Storage Collector

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **Azure Blob** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** 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. E.g., `azure_42-a`.

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** buttons to select one of these options:

- **Manual**: Use this default option to enter your Azure Storage connection string directly. Exposes a **Connection string** field for this purpose. (If left blank, Cribl 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.

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 for filtering and grouping in Cribl Stream. 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.

Last updated by: Dritan Bitincka

;

7.1.2. Filesystem/NFS

Cribl Stream supports collecting data from a locally mounted filesystem location that is available on all Worker Nodes.

Configuring a Filesystem Collector

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **Filesystem** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** to open the **Filesystem** > **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`.

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 for filtering and grouping in Cribl Stream. 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.

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

Environment: If you're using GitOps, 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.

Last updated by: Dritan Bitincka

;

7.1.3. 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 of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **Google Cloud Storage** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** 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`.

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** buttons 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 for filtering and grouping in Cribl Stream. 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

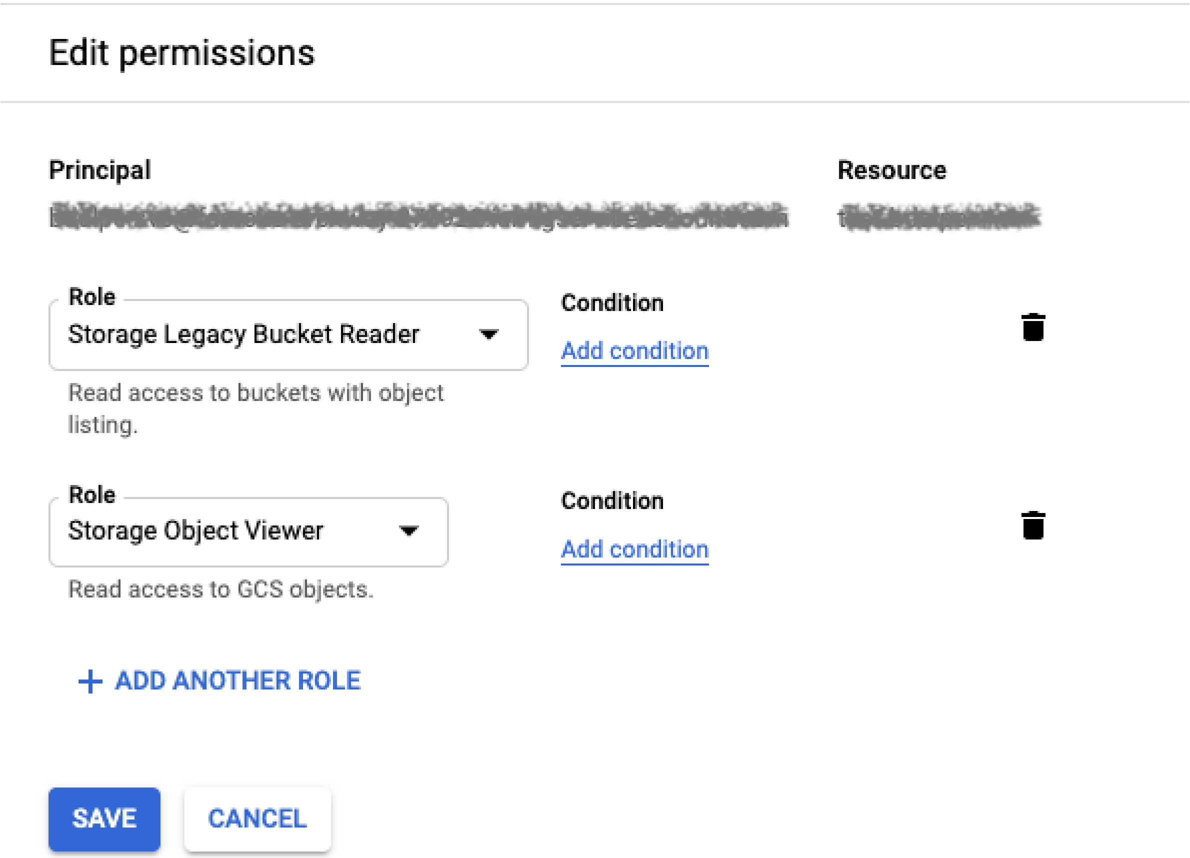
Your Google Cloud service account will need, at a minimum, roles and permissions that enable you to set up a new bucket or modify an existing one:

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

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.

Last updated by: Dritan Bitincka

;

7.1.4. 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.

Configuring a REST Collector

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 New** to open the **REST** > **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 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., `rest42json`.

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, etc. – 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)

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.

> 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`, or `POST with body`.

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}` ``.

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 (e.g., `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (e.g., `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 backticks (e.g., `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (e.g., `earliest`) 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 two additional fields:

- **Response attribute**: The attribute name in the response payload that contains next-page information. If you have multiple attributes with the same name, you can provide the path to the one you need, e.g., `foo.0.bar`.
- **Max pages**: The maximum number of pages to retrieve. Set to `0` to retrieve all pages.

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. E.g.: `/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. E.g.: `/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. E.g.: `/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. E.g.: `/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 buttons 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

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 Collect 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., `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

Login Secret Authentication

Selecting **Login (credentials secret)** exposes the following additional fields:

- **Login URL**: Endpoint 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 Collect 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., `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 – e.g., `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.
- **Token attribute**: Path to the token attribute in the login response body. Supports nested attributes.
- **Authorize expression**: JavaScript expression used to compute the Authorization header to pass in Discover and Collect calls. Uses `${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., `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 – e.g., `earliest` instead of `` `${earliest}` `` – 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 Collect 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., `earliest` instead of `` `${earliest}` `` – will be evaluated as strings.

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.

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 for filtering and grouping in Cribl Stream. 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`, 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., `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (e.g., `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 (e.g., `` `${earliest}` ``). Can also be a constant, enclosed in single quotes (`'earliest'`). Values without delimiters (e.g., `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), 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 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`, 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.

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 (etc.), 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)}` ``.

Response Errors

The Collector treats all non-200 responses from configured URL endpoints as errors. This includes 1xx, 3xx, 4xx, and 5xx responses.

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.

Last updated by: Dritan Bitincka

;

7.1.5. 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.

Incompatible Storage Classes

> Cribl Stream does **not** support data preview, collection, or replay from S3 Glacier or Deep Glacier storage classes, whose stated retrieval lags (variously minutes to 48 hours) cannot guarantee data availability when the Collector needs it.

Configuring an S3 Collector

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **S3** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** 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. E.g., `Attic42TreasureChest`.

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

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 presents 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.

- **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.

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.

Assume Role

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.)

Optional Settings

Region: S3 Region from which to retrieve data.

Path: Path, within the bucket, from which to collect data. Templating is supported (e.g., `/myDir/${host}/${year}/${month}/`). 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: S3 service endpoint. If empty, Cribl Stream will automatically construct the endpoint from the region.

Signature version: Signature version to use for signing S3 requests. Defaults to `v4`.

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.

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 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.`

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

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.

Last updated by: Dritan Bitincka

;

7.1.6. Script

Cribl Stream supports flexible data collection configured by your custom scripts.

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.

Configuring a Script Collector

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **Script** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** 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`.

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`.

> **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! 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. ☠️ ☠️

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

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.

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.

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`

Last updated by: Dritan Bitincka

;

7.1.7. 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 of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **Splunk Search** from the **Data Sources** page's tiles or the **Sources** left nav. Click **+ Add New** 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`.

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, use the buttons 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.

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.

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

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.

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.

Last updated by: Dritan Bitincka

;

7.1.8. 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 global ⚙ **Settings** (lower left) > **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 global ⚙ **Settings** (lower left) > **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 global ⚙ **Settings** (lower left) > **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.

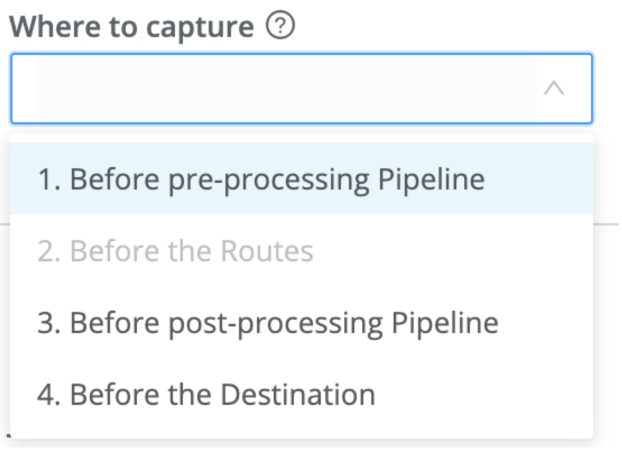
Preview Settings

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 slider 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.

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** to set start and end times like these:

- **Earliest** example values: `-1h`, `-42m`,`-42m@h`
- **Latest** example values: `now`, `-20m`, `+42m@h`

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 Filesystem 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).

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:

- `'Yy'`, for years
- `'mBbj'`, for months
- `'dj'`, for days
- `'HI'`, 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}/${_time:%d}/...`	`/my/path/2020/04/20/...`
`/my/path/${_time:year=%Y}/${_time:month=%m}/${_time:date=%d}/...`	`/my/path/year=2020/month=05/date`
`/my/path/${_time:%Y-%m-%d}/...`	`/my/path/2020-05-20/...`

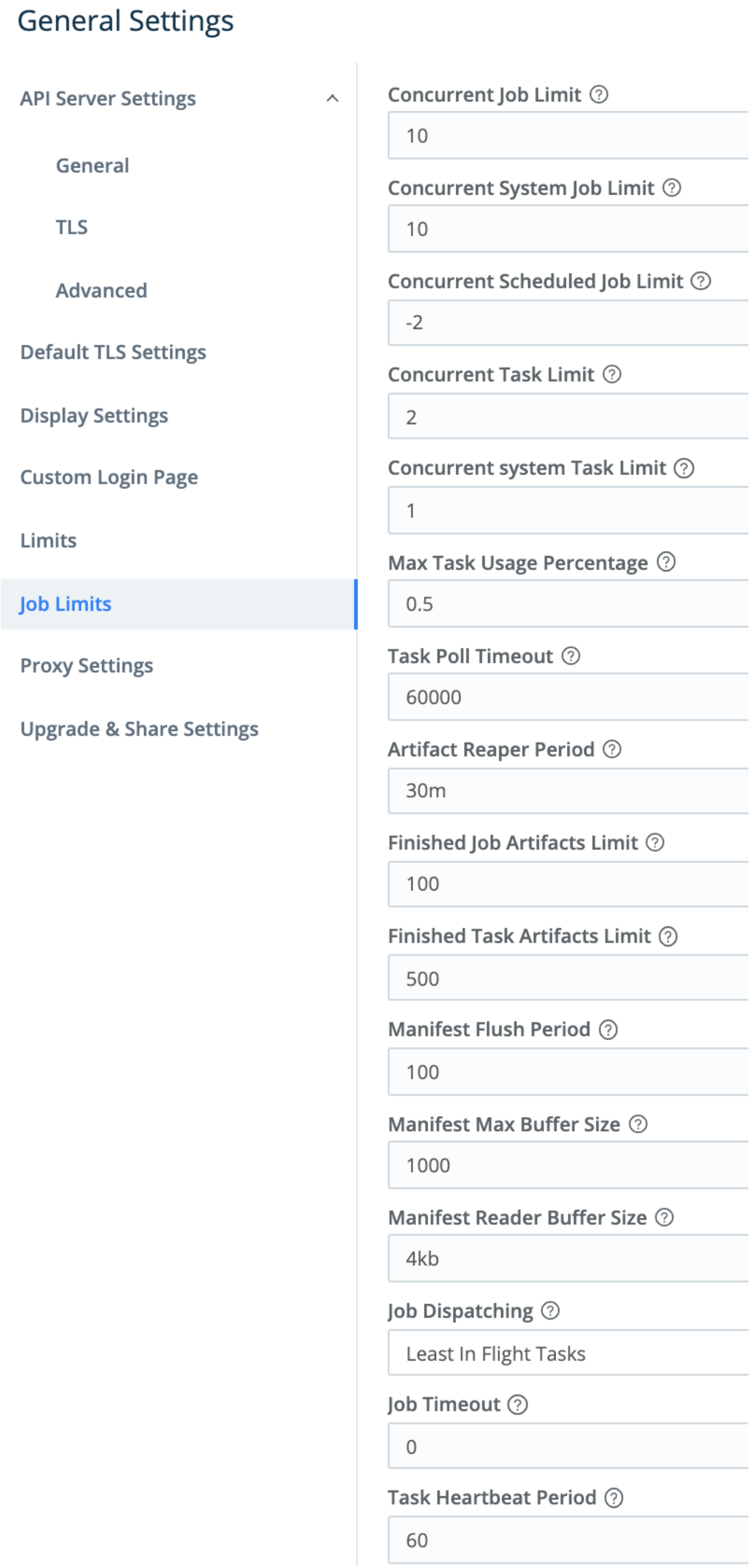
Last updated by: Dritan Bitincka

;

7.1.9. Job Limits

You can configure global limits that optimize the execution of all Collectors and scheduled jobs (including Cribl Stream system tasks).

The following limits are available at global ⚙ **Settings** (lower left) > **System** > **General Settings** > **Job Limits**:



Concurrent Job Limit: 10

Concurrent System Job Limit: 10

Concurrent Scheduled Job Limit: -2

Concurrent Task Limit: 2

Concurrent system Task Limit: 1

Max Task Usage Percentage: 0.5

Task Poll Timeout: 60000

Artifact Reaper Period: 30m

Finished Job Artifacts Limit: 100

Finished Task Artifacts Limit: 500

Manifest Flush Period: 100

Manifest Max Buffer Size: 1000

Manifest Reader Buffer Size: 4kb

Job Dispatching: Least In Flight Tasks

Job Timeout: 0

Task Heartbeat Period: 60

Job Limits settings

Limits Available

The following controls are available at global ⚙ **Settings** (lower left) > **General Settings > Job Limits**.

> In a distributed deployment, these limits are set on, and deployed from, the Leader. They are applied at the Worker Group level (except where noted), and trickle down to individual Worker Processes in the group. Task limits are applied at the Worker Process level.
>
> In a single-instance deployment, these limits are set on the single instance, and apply to all its Worker Processes.

Job Limits

Concurrent Job Limit: The total number of jobs that can run concurrently. Defaults to `10`.

> If you see jobs being skipped, this indicates that the **Concurrent Job Limit** for this Group has been reached or exceeded. Here, you need to increase this limit to reduce the number of skippable jobs. Note that, for resource-intensive jobs, this might trigger a need to deploy more Worker Nodes.

Concurrent System Job Limit: The total number of **system** jobs that can run concurrently. Defaults to `10`.

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`.

Task Limits

Concurrent Task Limit: The total number of tasks that a Worker Process can run concurrently. Defaults to `2`.

Concurrent System Task Limit: The number of system tasks that a Worker Process can run concurrently. Defaults to `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).

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`.

Finished Task Artifacts Limit: Maximum number of finished task artifacts to keep on disk, per job, on each Worker Node. Defaults to `500`.

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.

Manifest Max Buffer Size: The maximum number of tasks that the task manifest can hold in memory before flushing to disk. Defaults to `1,000`.

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.

Last updated by: Dritan Bitincka

;

7.2. Amazon

7.2.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Amazon** > **Firehose**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Amazon** > **Firehose**. Next, click **+ Add New** to open an **Amazon Firehose** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Address to bind on. Defaults to 0.0.0.0 (all addresses).

Port: Enter the port number to listen on.

Authentication Settings

Auth tokens: Shared secrets to be provided by any client (Authorization: \<token>). Click **Generate** to create a new secret. If empty, unauthenticated access **will be permitted**.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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:

- `__firehoseArn`
- `__firehoseEndpoint`
- `__firehoseReqId`
- `__firehoseToken`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.
- `__inputId`
- `__srcIpPort` – See details below.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Kinesis Firehose client sending data to this Source.

When any proxies (including load balancers) lie between the Kinesis Firehose client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** is set to `Yes`, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Limitations/Troubleshooting

If you set the optional `IntervalInSeconds` and/or `SizeInMBs` parameters in the Kinesis Firehose `BufferingHints` API, beware of selecting extreme values (toward the ends of the API's supported ranges). These can send more bytes than Cribl Stream can buffer, causing Cribl Stream to send HTTP 500 error responses to Kinesis Firehose.

;

7.2.2. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Amazon** > **Kinesis**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Amazon** > **Kinesis**. Next, click **+ Add New** to open an **Amazon Kinesis** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Kinesis Stream Source definition.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method:

- **Auto**: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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 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

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.

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.

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.)

Environment: If you're using GitOps, 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`

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.

;

7.2.3. Amazon S3

Cribl Stream supports receiving data from Amazon S3 buckets, using event notifications through 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.

Incompatible Storage Classes

> Cribl Stream does **not** support data ingestion from buckets saved to S3's Glacier or Deep Glacier storage classes – whose stated retrieval lags (variously, minutes to 48 hours) cannot guarantee data availability.

Configuring Cribl Stream to Receive Data from Amazon S3

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Amazon** > **S3**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Amazon** > **S3**. Next, click **+ Add New** to open an **Amazon S3** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this S3 Source definition.

Queue: The name, URL, or ARN of the SQS queue to read events from. When specifying a non-AWS URL, you must use the format: `{url}/<queueName>`. (E.g., `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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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

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

Endpoint: S3 service endpoint. If empty, defaults to AWS's region-specific endpoint. Otherwise, used to point to an S3-compatible endpoint.

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): The amount of time to wait for events before polling again. Acceptable values: `1` (default) to `20`. Short durations increase the number (and thus cost) of requests sent to AWS. 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. (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 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.

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`
- `__source`

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": "example-2020-04-20",
 "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`

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. Which means, **each** Worker Process, in **each** Cribl Stream Worker Node, will run 1 receiver consuming 1 message (i.e., 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 no problems with the CLI, see if the AWS CLI proxy is in use. Check with your administrator for details.

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**.

;

7.2.4. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Amazon** > **SQS**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Amazon** > **SQS**. Next, click **+ Add New** to open an **Amazon SQS** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this SQS Source definition.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

Auto

This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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.

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): The amount of time to wait for events before polling again. Acceptable values: `1` (default) to `20`. Short durations increase the number (and thus cost) of requests sent to AWS. 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:

- `__inputId`
- `__sqsSysAttrs`

The `_sqsSysAttrs` field can take on the following properties, which are reported to Cribl Stream from SQS:

- `__sqsSysAttrs.ApproximateFirstReceiveTimestamp` : Returns the time (epoch time in milliseconds) the message was first received from the queue.
- `__sqsSysAttrs.ApproximateReceiveCount` : Returns the number of times a message has been received from the queue without being deleted.
- `__sqsSysAttrs.SenderId` : For an IAM user, returns the IAM user ID (e.g.: `ABCDEFGHI1JKLMNOPQ23R`). For an IAM role, returns the IAM role ID (e.g.: `ABCDE1F2GH3I4JK5LMNOP:i-a123b456`).
- `__sqsSysAttrs.SentTimestamp` : Returns the time (epoch time in milliseconds) the message was sent to the queue.
- `__sqsSysAttrs.MessageDeduplicationId` : Returns the value provided by the producer that calls the `SendMessage` action.
- `__sqsSysAttrs.MessageGroupId` : Returns the value provided by the producer that calls the `SendMessage` action – messages with the same `MessageGroupId` are returned in sequence.
- `__sqsSysAttrs.SequenceNumber` : Returns the sequence-number value provided by Amazon SQS.

- `__sqsSysAttrs.AWSTraceHeader`: Returns the AWS X-Ray trace header string.

For background on these message properties, see AWS' ReceiveMessage > Request Parameters documentation.

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)

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.

;

7.3. Azure

7.3.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Azure** > **Blob Storage**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Azure** > **Blob Storage**. Next, click **+ Add New** to open an **Azure Blob Storage** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Azure Blob Storage Source definition.

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** buttons to select one of these options:

- **Manual**: Use this default option to enter your Azure Storage connection string directly. Exposes a **Connection string** field for this purpose. (If left blank, Cribl 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=<your-account-key>`

A fictitious example, using Microsoft's recommended HTTPS option, is: `DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32`

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 for filtering and grouping in Cribl Stream. 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:

- `__inputId`
- `__source`

> 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.

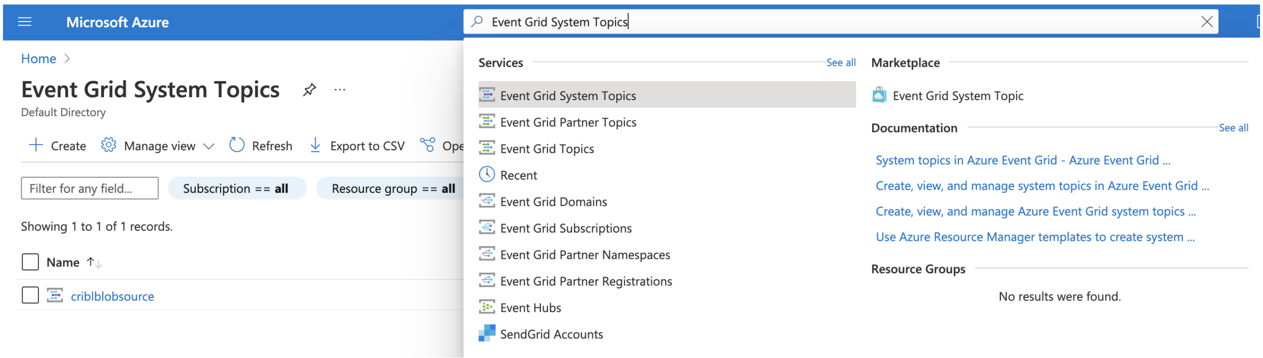
2. Create a queue.

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, tart at **Event Grid System Topics**:



Microsoft Azure

Event Grid System Topics

Home >

Event Grid System Topics

Default Directory

+ Create   Manage view   Refresh   Export to CSV   Ope

Filter for any field...   Subscription == **all**   Resource group == **all**

Showing 1 to 1 of 1 records.

☐ Name ↑↓

☐ criblblobsource

Services   See all

- Event Grid System Topics
- Event Grid Partner Topics
- Event Grid Topics
- Recent
- Event Grid Domains
- Event Grid Subscriptions
- Event Grid Partner Namespaces
- Event Grid Partner Registrations
- Event Hubs
- SendGrid Accounts

Marketplace

- Event Grid System Topic

Documentation   See all

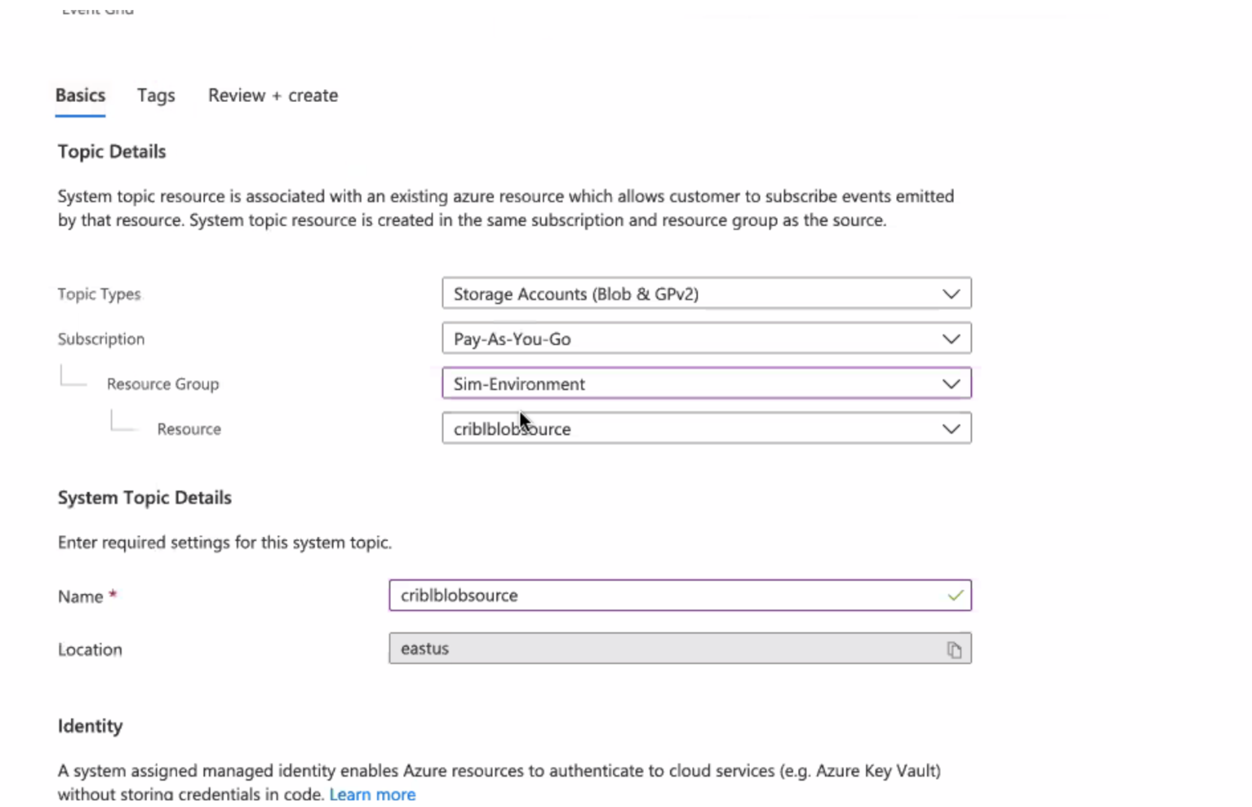
- System topics in Azure Event Grid - Azure Event Grid ...
- Create, view, and manage system topics in Azure Event Grid ...
- Create, view, and manage Azure Event Grid system topics ...
- Use Azure Resource Manager templates to create system ...

Resource Groups

No results were found.

Azure portal > System topics

Select **+Create** to create a new system topic, then set the **Topic Type** to **Storage Account (Blob)**:



Event Grid

Basics   Tags   Review + create

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

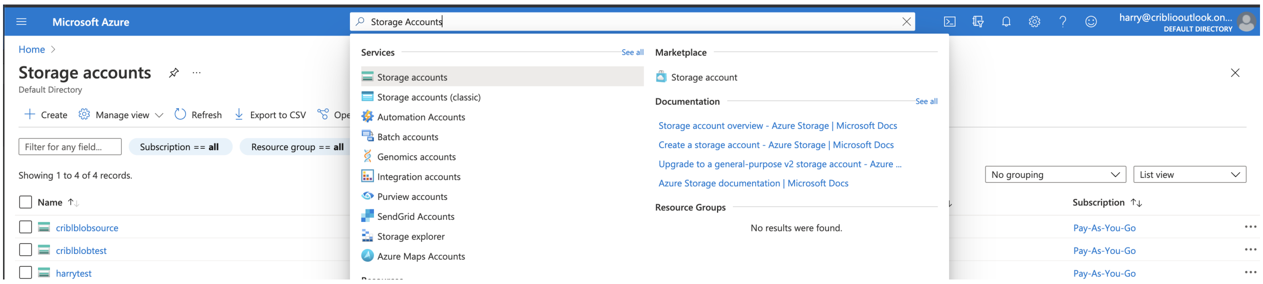
Creating a system topic

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.)

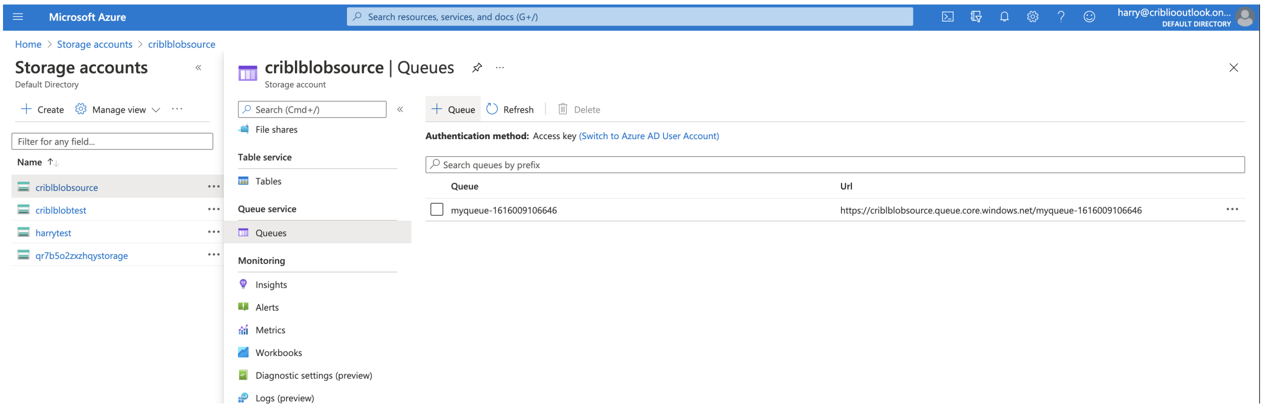
2. Create Storage Queue

Next, navigate to your storage account to create a queue.



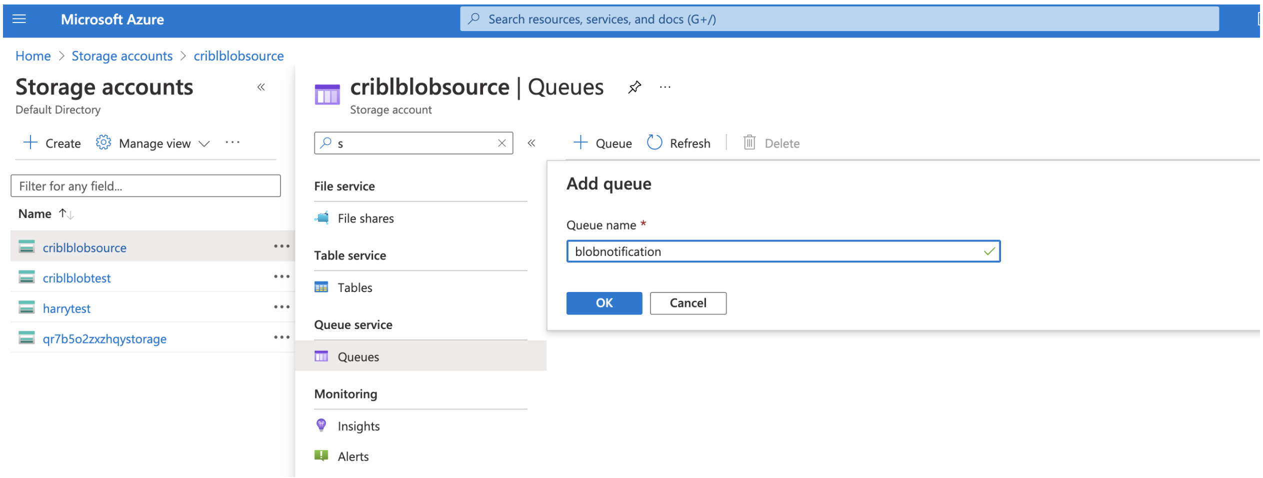
Accessing your storage account

Select the storage account for which you would like to set up notifications. Then, in the submenu, select **Queue service > Queues**:



Accessing queues

Select **Create queue**, and give the queue a name that is meaningful to you.

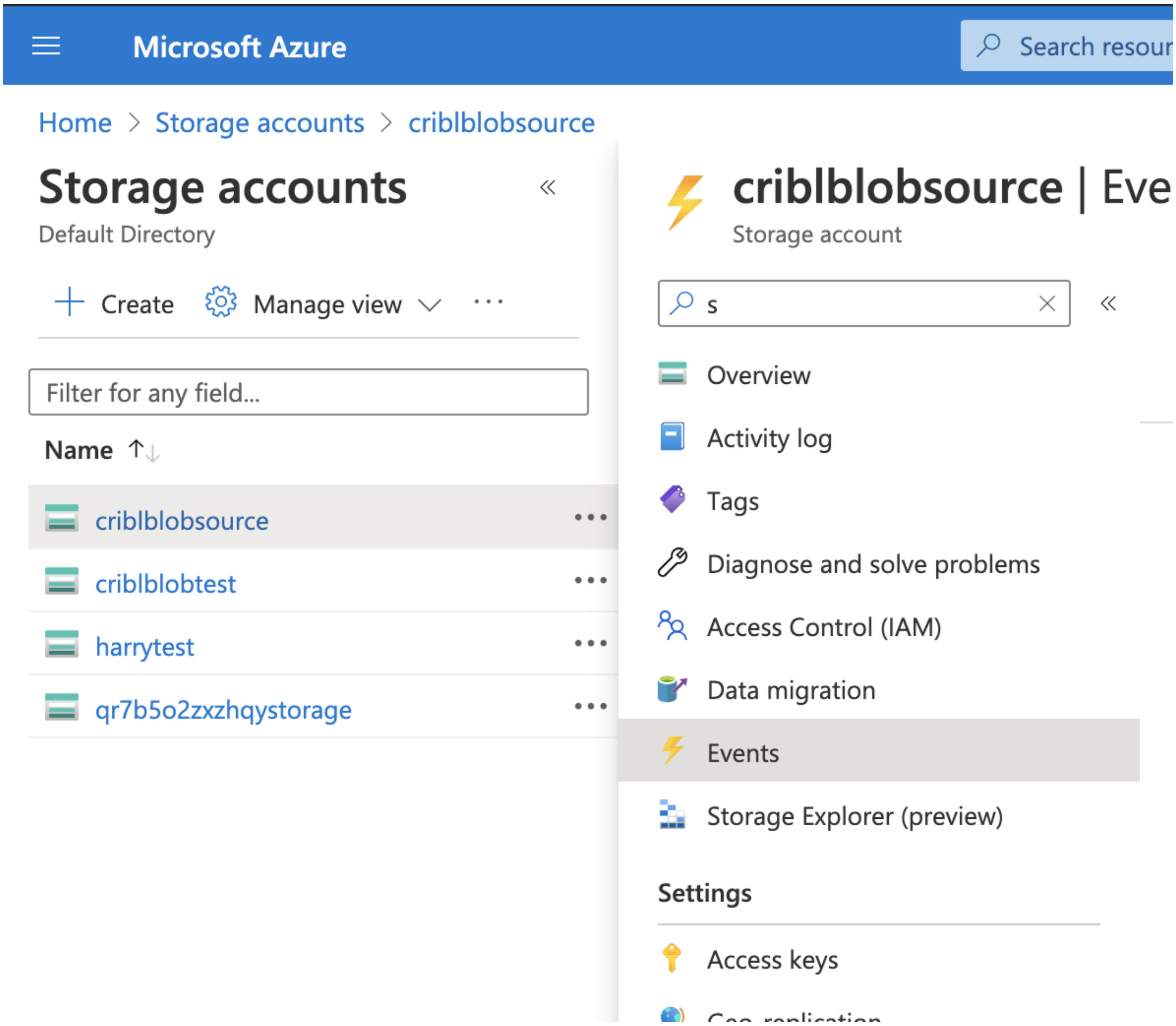


Queue name * blobnotification

Adding a queue

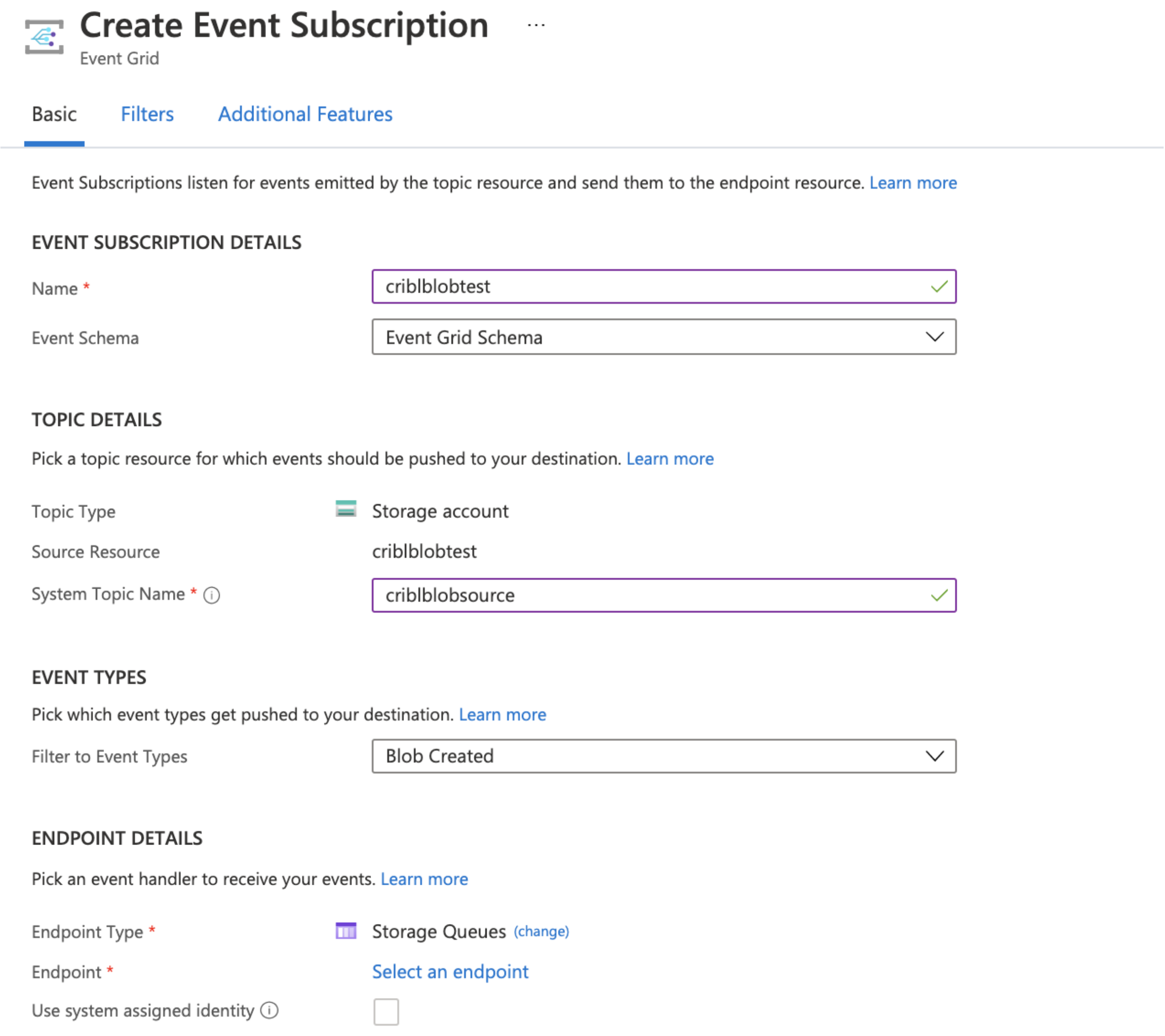
3. Configure Storage Account Notifications

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

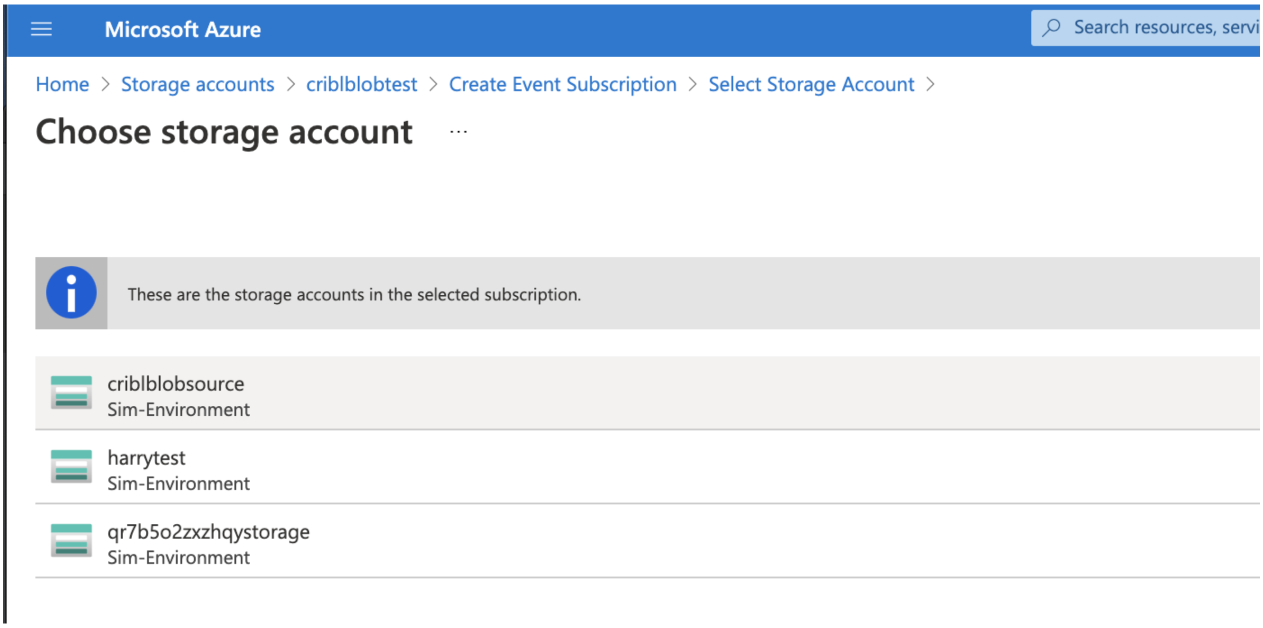
Use system assigned identity ⓘ □

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
Search resources, servi
Home > Storage accounts > criblblobtest > Create Event Subscription > Select Storage Account >

Choose storage account ...

These are the storage accounts in the selected subscription.

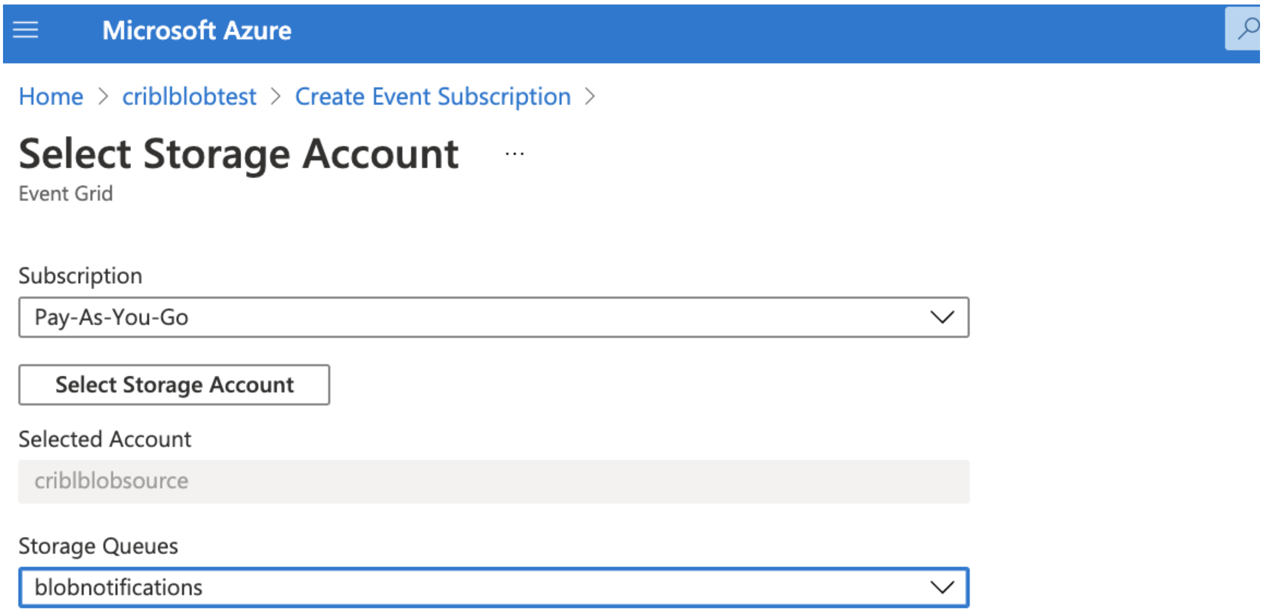
criblblobsource
Sim-Environment

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.



Select Storage Account ...

Event Grid

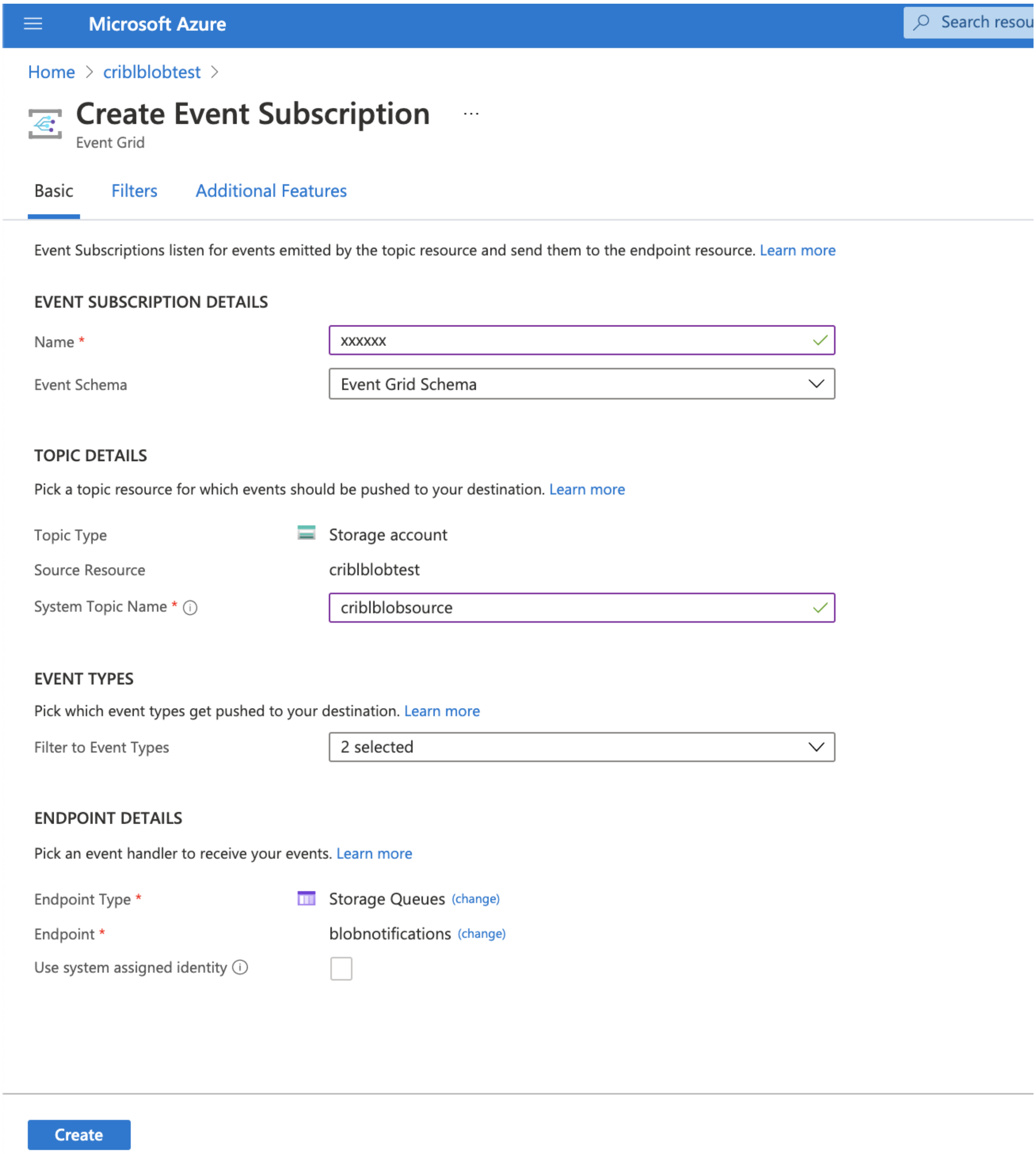
Subscription: Pay-As-You-Go

Selected Account: criblblobsource

Storage Queues: blobnotifications

Selecting the storage account

To complete the process, click **Create**.



Create Event Subscription

Event Grid

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.

;

7.3.2. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Azure** > **Event Hubs**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Azure** > **Event Hubs**. Next, click **+ Add New** to open an **Azure Event Hubs** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this source definition.

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 two 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.
> - You should configure a **separate** Azure Event Hubs Source for each Group:Topic pair whose events you want to subscribe to.

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 for filtering and grouping in Cribl Stream. 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 `No` – and for Event Hubs, must always be disabled.

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. Currently, `PLAIN` is the only mechanism supported for Event Hubs Kafka brokers.

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 uses an Azure Event Hubs connection string in this format:

```
Endpoint=sb://<FQDN>/;SharedAccessKeyName=<your-shared-access-key-name>;SharedAccessKey=<your-shared-access-key-value>
```

A fictitious example is:

```
Endpoint=sb://dummynamespace.servicebus.windows.net/;SharedAccessKeyName=DummyAccessKeyName;SharedAccessKey=5dOntTRytoC24opYThisAsit3is2B+OGY1US/fuL3ly=
```

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.

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.

;

7.4. Google Cloud

7.4.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Google Cloud** > **Pub/Sub**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Google Cloud** > **Pub/Sub**. Next, click **+ Add New** to open a **Google Cloud Pub/Sub** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Pub/Sub Source definition.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons 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.

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`

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.

;

7.5. Kafka

7.5.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Kafka**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Kafka**. Next, click **+ Add New** to open a **Kafka** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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, Cribl suggests subscribing each Kafka Source to only one topic. This prevents excessive rebalancing. If you want to subscribe to multiple topics, consider creating a dedicated Kafka Source for each one.

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 for filtering and grouping in Cribl Stream. 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): 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 accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Authentication

This section governs SASL (Simple Authentication and Security Layer) authentication to use when connecting to brokers.

Enabled: Defaults to `No`. When toggled to `Yes`:

SASL mechanism: Use this drop-down to select the SASL authentication mechanism to use. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these authentication mechanisms, select one of the following buttons:

Manual: Displays **Username** and **Password** fields to enter your Kafka credentials directly.

Secret: This option exposes a **Credentials secret** drop-down in which you can select a stored text secret that references your Kafka credentials. A **Create** link is available to store a new, reusable secret.

GSSAPI/Kerberos

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the key table file for the authentication principal.

Principal: Enter the authentication principal, e.g.: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, e.g.: `kafka`.

Schema Registry

This section governs Kafka Schema Registry authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (E.g., `http://<hostname>:8081`.)

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.

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.

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.

;

7.5.2. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Confluent Cloud**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Confluent Cloud**. Next, click **+ Add New** to open a **Confluent Cloud** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

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, Cribl suggests subscribing each Confluent Cloud Source to only one topic. This prevents excessive rebalancing. If you want to subscribe to multiple topics, consider creating a dedicated Confluent Cloud Source for each one.

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 for filtering and grouping in Cribl Stream. 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): 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 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

Enabled: Defaults to `No`. When toggled to `Yes`, all the settings in this section are required.

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use. Currently, `PLAIN` is the only mechanism supported for Confluent Kafka brokers.

Username: The username for authentication. For Confluent, this should always be `$ConnectionString`.

Authentication method: Use the buttons to select one of these options:

- **Manual**: Use this default option to enter your Confluent connection string. Exposes a **Password** field for this purpose.
- **Secret**: This option exposes a **Connection string (text secret)** drop-down, in which you can select a stored secret that references a Confluent 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.

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)**: 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.

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.

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.

;

7.6. Office 365

7.6.1. Office 365 Activity

Cribl Stream supports receiving data from the Office 365 Management Activity API. This facilitates analyzing actions and events on Azure Active Directory, Exchange, and SharePoint, along with global auditing and Data Loss Prevention data.

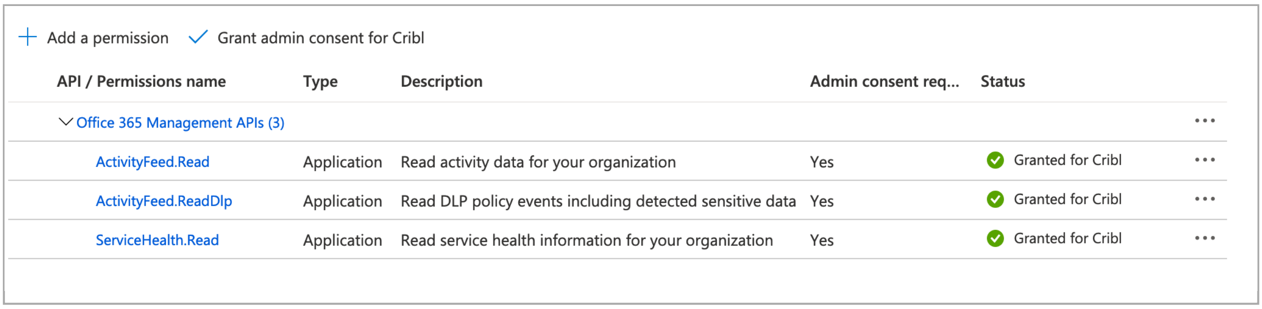
> Type: **Pull** | TLS Support: **YES** | Event Breaker Support: **YES**

TLS is enabled via the HTTPS protocol on this Source's underlying REST API.

Azure AD Permissions

In Azure Active Directory, 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.



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

In the **QuickConnect UI**: Click **+ New Source**, or click **+ Add Source** beside Sources. From the resulting drawer's tiles, select [**Pull** >] **Office 365** > **Activity**. Next, click **+ Add New** to open a **New Source** modal that provides the following options and fields.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Office 365** > **Activity**. Next, click **+ Add New** to open a **Office 365 Activity** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Office 365 Activity definition.

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. This is typically `Enterprise` or `GCC Government Plan.`

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 for filtering and grouping in Cribl Stream. 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.

Advanced Settings

Keep Alive Time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `60`.

Worker timeout (periods): The number of Keep Alive Time periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

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.

Environment: If you're using GitOps, 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`

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 Azure AD, generate a Client Secret, and update th
and client secret here
# Get the tenant GUID from Properties | Directory ID under the Azure Active Directory 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=$AppID;client_secret=$Cl
$oauth = Invoke-RestMethod -Method Post -Uri $loginURL/$TenantID/oauth2/token?api-versi
Body $body
$headerParams = @{'Authorization'="$($oauth.token_type) $($oauth.access_token)"}

Invoke-WebRequest -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/list"

Invoke-WebRequest -Method Post -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/start?
contentType=Audit.AzureActiveDirectory"
Invoke-WebRequest -Method Post -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/start?contentType=Audit.Excha
Invoke-WebRequest -Method Post -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/start?contentType=Audit.Share
Invoke-WebRequest -Method Post -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/start?contentType=Audit.Gener
Invoke-WebRequest -Method Post -Headers $headerParams -Uri
"$resource/api/v1.0/$TenantID/activity/feed/subscriptions/start?contentType=DLP.All"
```

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&client_id=00000000-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":"1622089429","not_before":"1622085529","resource":"https://manage.office.com","access_token":"eyJ0...long 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.

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.

;

7.6.2. 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 or OAuth Secret authentication, your Office 365 setup must include at least one role with the corresponding required permissions:

1. In the Azure portal, create an Azure AD 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

For detailed steps, see Microsoft's Assign Azure AD Roles to Users topic.

Configuring Cribl Stream to Receive Office 365 Message Trace Data

In the **QuickConnect UI**: Click **+ New Source**, or click **+ Add Source** beside Sources. From the resulting drawer's tiles, select [**Pull** >] **Office 365** > **Message Trace**. Next, click **+ Add New** to open a **New Source** modal that provides the following options and fields.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Office 365** > **Message Trace**. Next, click **+ Add New** to open a **Office 365 Message Trace** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Office 365 Message Trace definition.

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, or OAuth (text secret). The default is **OAuth**. Both 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

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 Azure Active Directory.
- **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 Azure Active Directory.
- **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.

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 for filtering and grouping in Cribl Stream. 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.

Advanced Settings

Keep Alive time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `60`.

Worker timeout (periods): The number of Keep Alive Time periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

Timeout (secs): Maximum time to wait for an individual Message Trace API request to complete. Defaults to `600` seconds (10 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.

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.

;

7.6.3. Office 365 Services

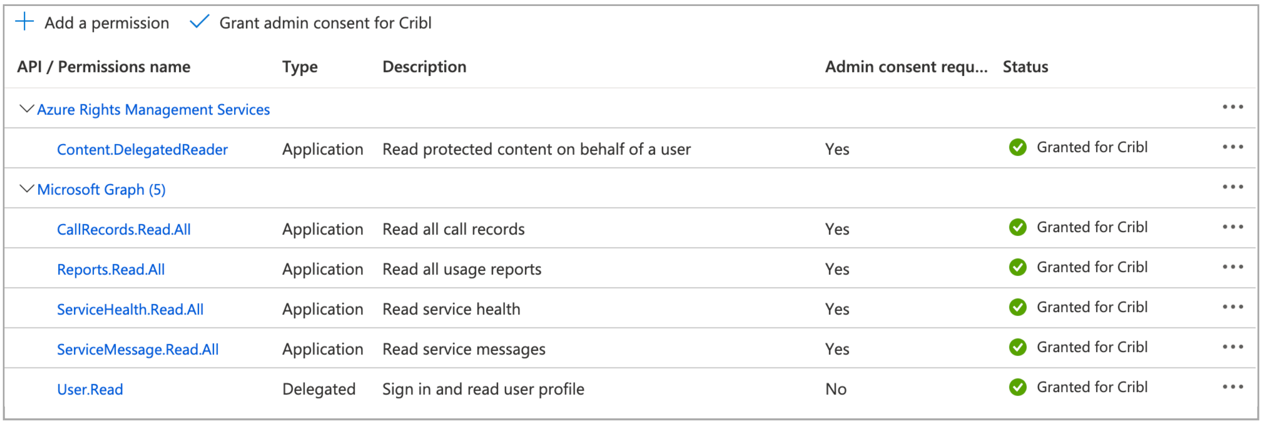
Cribl Stream supports receiving data from the Microsoft Graph service communications API. This facilitates analyzing the status and history of service incidents on multiple Microsoft cloud services, along with associated incident and Message Center communications. For details, see Microsoft's Overview 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.

Azure AD Permissions

In Azure Active Directory, 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`



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

In the **QuickConnect UI**: Click **+ New Source**, or click **+ Add Source** beside Sources. From the resulting drawer's tiles, select [**Pull** >] **Office 365** > **Services**. Next, click **+ Add New** to open a **New Source** modal that provides the following options and fields.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Office 365** > **Services**. Next, click **+ Add New** to open a **Office 365 Services** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Office 365 Services definition.

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

Content Types: See the Content Types section below.

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.

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.

Advanced Settings

Keep Alive Time (seconds): How often Workers should check in with the scheduler to keep their job subscription alive. Defaults to `60`.

Worker timeout (periods): The number of Keep Alive Time periods before an inactive Worker will have its job subscription revoked. Defaults to `3`.

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.

Environment: If you're using GitOps, 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.

;

7.7. Prometheus

7.7.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Prometheus** > **Scraper**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The drawer will now provide the following options and fields.

From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Prometheus** > **Scraper**. Next, click **+ Add New** to open a **Prometheus Scraper** > **New Source** modal that provides the following options and fields.

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.

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 **Assumre 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`. For further options, see Target Discovery for DNS.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication (Prometheus)

Use the **Authentication Method** buttons to select one of these authentication options for Prometheus:

- **Manual**: In the resulting **Username** and **Password** fields, enter Basic authentication credentials corresponding to your Prometheus targets.

- **Secret**: This option exposes a **Secret** drop-down, in which you can select a stored secret that references your credentials described above. The secret can reside in Cribl 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.

- **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.

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. 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.

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.

;

7.7.2. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Prometheus** > **Remote Write**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Prometheus** > **Remote Write**. Next, click **+ Add New** to open a **Prometheus Remote Write** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: Enter the port number to listen on..

Remote Write API endpoint: Enter the absolute path on which to listen for Prometheus requests. Defaults to `/write`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/write`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header, or click **Generate** if you need a new token.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or click **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials. Click **Generate** if you need a new password.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or click **Create** to configure a new secret.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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.

Keep alive timeout (seconds): Maximum time to keep a socket connection open to wait for additional data, after the last response was sent. When the incoming request frequency is high, increase this from the default `5` seconds, to avoid creating a new connection per request. (By default, Prometheus will attempt to keep connections open for up to 5 minutes.)

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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.

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`.

;

7.7.3. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Grafana**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Grafana**. Next, click **+ Add New** to open a **Grafana** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: Enter the port number to listen on.

Optional Settings

Remote Write API endpoint: Absolute path on which to listen for Grafana Agent's remote write requests. Defaults to `/api/prom/push`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/api/prom/push`.

Logs API endpoint: Absolute path on which to listen for Loki logs requests. Defaults to `/loki/api/v1/push`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/loki/api/v1/push`.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

The **Authentication** tab provides separate **Loki** and **Prometheus** sections, enabling you to configure these inputs separately. The two sections provide identical options.

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Enter the bearer token that must be included in the authorization header.
- **Auth token (text secret)**: Provide an HTTP token referenced by a secret. Select a stored text secret in the resulting drop-down, or click **Create** to configure a new secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or click **Create** to configure a new secret.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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.

Keep alive timeout (seconds): Maximum time to keep a socket connection open to wait for additional data, after the last response was sent. When the incoming request frequency is high, increase this from the default `5` seconds, to avoid creating a new connection per request. (By default, Grafana Agent's embedded Prometheus instance will attempt to keep connections open for up to 5 minutes.)

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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`.

;

7.7.4. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Loki**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Loki**. Next, click **+ Add New** to open a **Loki** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: Enter the port number to listen on.

Logs API endpoint: Absolute path on which to listen for Loki logs requests. Defaults to `/loki/api/v1/push`, which will (in this example) expand as: `http://<your-upstream-URL>:<your-port>/loki/api/v1/push`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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 matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal

characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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` – Will contain all the labels found in each event's corresponding Loki stream.
- `__srcIpPort` – See details below.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Loki client sending data to this Source.

When any proxies (including load balancers) lie between the Loki client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** is set to `Yes`, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

;

7.8. Splunk

7.8.1. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Splunk** > **HEC**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Splunk** > **HEC**. Next, click **+ Add New** to open a **Splunk HEC** > **New Source** modal that provides the following options and fields.

> 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.

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.

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 both JSON events via `/event` and raw events via `/raw`. See the examples below.

Optional Settings

Allowed Indexes: List the values allowed in the HEC event index field. Allows wildcards. Leave blank to skip validation.

Splunk HEC acks: Whether to enable Splunk HEC acknowledgments. Defaults to `No`. Some sources may require HEC acks to be enabled and, as a result, may keep TCP connections open while waiting for an ack. This behavior can exhaust available file descriptors. Cribl does not maintain a comprehensive list of such sources. Refer to your source's documentation for more information.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the

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

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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), 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 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.
>
> E.g., consider the following expression: `` `${__e['index'] || 'myIndex'}` `` Its L->R and OR logic specifies: If an inbound event includes an `index` field, use that field's value. Otherwise, fall back to the `myIndex` constant defined in this expression.
>
> Fields here are evaluated and applied **after** any fields specified in the Auth Tokens section.

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Auth Tokens

If empty (the default), the Splunk HEC Source will permit client access without an auth token. To generate and/or configure tokens, click **+ Add Token**, which exposes the following fields:

Token: Shared secret to be provided by any client (Authorization: <token>). Click **Generate** to create a new secret. If empty, unauthenticated access **will be permitted**.

Description: Optional description for this token.

Fields: Fields to add to events referencing this token. Each field is a **Name**/**Value** pair.

> Fields specified on the **Auth Tokens** tab will normally override fields of the same name in events. But you can specify that fields in events should override these fields' values.
>
> E.g., consider the following expression: `` `${__e['index'] || 'myIndex'}` `` Its L->R and OR logic specifies: If an inbound event includes an `index` field, use that field's value. Otherwise, fall back to the `myIndex` constant defined in this expression.
>
> Fields here are evaluated and applied **before** any fields specified in the Fields section.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle this to `Yes` to add request headers to events, in the `__headers` field.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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`
- `__srcIpPort` – See details below.

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.)

Format and Endpoint Examples

Splunk HEC to Cribl Stream

- Configure Cribl Stream to listen on port `10080` 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

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"this is a sample event ", "host":"myHost",
"source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42'
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'

# Multiple Events
curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42'
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}{"event":"this is a sample event 2",
"host":"myHost", "source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

# Metrics Events
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"metric", "host":"myHost", "fields":
{"_value":3850,"metric_name":"kernel.entropy_avail"}}'

curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
myToken42' -d '{"host":"myHost", "fields":
{"_value":3850,"metric_name":"kernel.entropy_avail"}}'

# Send the auth token as a query parameter, with no additional configuration

curl -k "http://<myCriblHost>:10080/services/collector/event?token=mToken42" -d
'{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

Splunk HEC to Cribl Cloud

- Navigate to Cribl Cloud's Splunk HEC Source > **Auth Tokens** tab.
- Copy your token out of the **Token** field.
- From the command line, use `https`, your Cribl.Cloud portal's **Ingest Endpoint** and port, and the token's value:

```
curl -k "https://in.logstream.<tenant-ID>.cribl.cloud:8088/services/collector" \
    -H "Authorization: <token_value>" \
    -d '{"event": "Goats are better than ponies."}{"event": "Goats are better
climbers."}{"event": "Goats are great yoga buddies.", "nested": {"horns": "Two is
better than one!"}}'
```

;

7.8.2. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **Splunk Search**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **Splunk Search**. Next, click **+ Add New** to open a **Splunk Search** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Splunk Search Source definition.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

In the **Authentication** tab, use the buttons 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.

- **Manual**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication 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.

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.

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`.

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.

;

7.8.3. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Splunk** > **Splunk TCP**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Splunk** > **Splunk TCP**. Next, click **+ Add New** to open a **Splunk TCP** > **New Source** modal that provides the following options and fields.

> 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.

Address: Enter hostname/IP to listen for Splunk data. E.g., `localhost` or `0.0.0.0`.

Port: Enter port number.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

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.

Environment: If you're using GitOps, 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`
- `__srcIpPort`
- `__source`

Configuring a Splunk Forwarder

To configure a Splunk forwarder (UF, HF) use the following sample outputs.conf stanzas:

outputs.conf (on-prem)   outputs.conf (Cribl Cloud)

```
[tcpout]
disabled = false
defaultGroup = cribl, <optional_clone_target_group>,

[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ...
sendCookedData=true
# As of Splunk 6.5, using forceTimebasedAutoLB is no longer recommended. Ensure this
is left at default for UFs
# forceTimebasedAutoLB = false
negotiateProtocolLevel = 0
```

> To avoid data loss, Cribl recommends that you use the `negotiateProtocolLevel = 0` setting shown above. Depending on your environment, enabling `negotiateProtocolLevel` could cause Cribl to not accept data from the forwarder.

;

7.9. Internal

7.9.1. 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 and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**

Configuring Cribl Stream to Generate Sample Data

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **Datagen**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Datagen**. Next, click **+ Add New** to open a **Datagen** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Datagens: List of datagens.

- **Data generator file**: Name of the datagen file.
- **Events per second per Worker Node**: Maximum number of events to generate per second, per Worker Node/Edge Node. Defaults to `10`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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`

;

7.9.2. 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: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**
>
> Cribl Cloud instances omit this Source, because Cribl manages these instances' uptime and diagnostics on your behalf.

Scope

In distributed mode, only Worker Process internal logs can be processed through this Source. (Logs on the Leader remain on the Leader, since the Leader Node is not part of any processing path.)

In both distributed and single-instance mode, this Source 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.

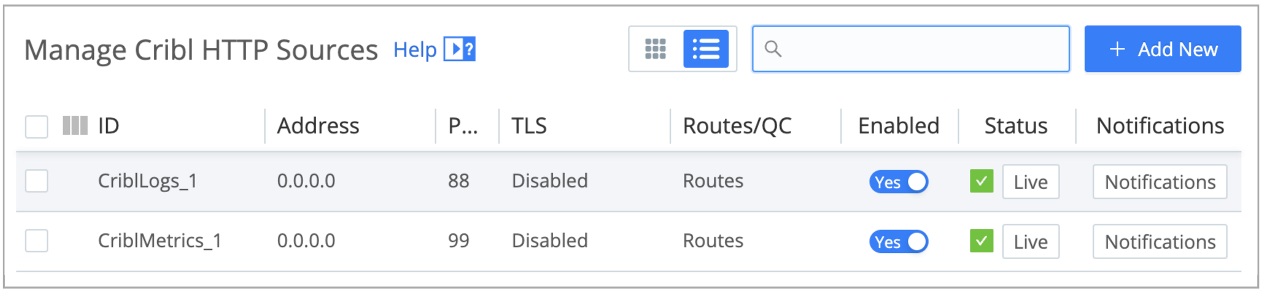
Configuring Cribl Internal Logs/Metrics as a Data Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl Internal**. Next, click **Select Existing**.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Cribl Internal**.

Next, in either UI: On the **CriblLogs** and/or the **CriblMetrics** row, slide the **Enabled** slider to `Yes`. Confirm your choice in the resulting message box.


Manage Cribl HTTP Sources

☐	ID	Address	P...	TLS	Routes/QC	Enabled	Status	Notifications
☐	CriblLogs_1	0.0.0.0	88	Disabled	Routes	Yes	✓ Live	Notifications
☐	CriblMetrics_1	0.0.0.0	99	Disabled	Routes	Yes	✓ Live	Notifications

Cribl Internal Sources – click to configure

CriblLogs – General Settings

Enabled: This duplicates the parent page's **Enabled** slider. Keep it at `Yes` to enable Cribl logs as a Source.

Input ID: Enter a unique name to identify this CriblLogs Source definition.

CriblLogs – Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

> See also Common Settings below.

CriblMetrics – General Settings

Enabled: This duplicates the parent page's **Enabled** slider. Keep it at `Yes` to enable Cribl metrics as a Source.

Input ID: Enter a unique name to identify this CriblMetrics Source definition.

CriblMetrics – Optional Settings

Metric name prefix: Enter an optional prefix that will be applied to metrics provided by Cribl 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 for filtering and grouping in Cribl Stream. 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 slider described above.

Omitting `sourcetype`

You can easily drop the `sourcetype` attribute from metrics events, leaving only `event_sourcetype`. This will prevent duplicate `sourcetype` events from being routed to Destinations.

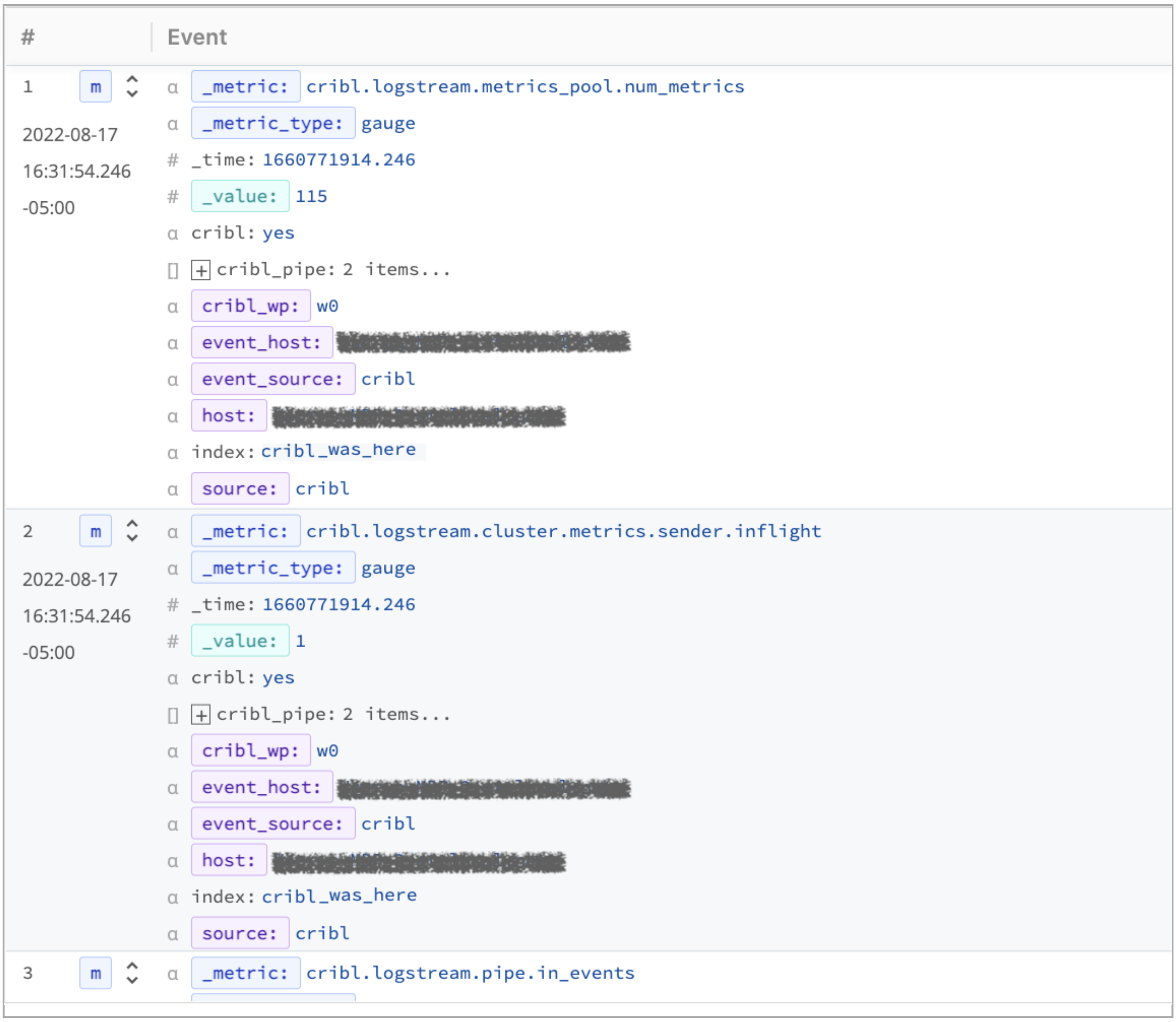
To do this: In the same `cribl_metrics_rollup` pre-processing Pipeline (or a clone) that you attach to your Source, enable the final Eval Function, which applies this **Filter** expression to remove the `sourcetype` field: `_metric && _metric.startsWith('cribl.logstream.sourcetype.')`

Duplicated Fields/Dimensions

The CriblMetrics Source operates on metrics that 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                 _metric: cribl.logstream.metrics_pool.num_metrics
2022-08-17           _metric_type: gauge
16:31:54.246         _time: 1660771914.246
-05:00               _value: 115
                     cribl: yes
                     cribl_pipe: 2 items...
                     cribl_wp: w0
                     event_host: [redacted]
                     event_source: cribl
                     host: [redacted]
                     index: cribl_was_here
                     source: cribl
2  m                 _metric: cribl.logstream.cluster.metrics.sender.inflight
2022-08-17           _metric_type: gauge
16:31:54.246         _time: 1660771914.246
-05:00               _value: 1
                     cribl: yes
                     cribl_pipe: 2 items...
                     cribl_wp: w0
                     event_host: [redacted]
                     event_source: cribl
                     host: [redacted]
                     index: cribl_was_here
                     source: cribl
3  m                 _metric: cribl.logstream.pipe.in_events
```

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.

;

7.9.3. Cribl HTTP

The internal Cribl HTTP 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.

> Type: **System and Internal** | TLS Support: **YES** | Event Breaker Support: **No**

You might choose this Source over the Cribl TCP Source in certain circumstances, such as when a firewall or proxy blocks raw TCP egress. In single-instance mode or for testing, you can substitute the Raw HTTP/S Source. (However, this substitution will not provide the single-billing benefits described in the next section.)

How It Works

You can use the Cribl HTTP Source to send data between Workers. In a hybrid Cloud deployment, using this Source ensures that you're billed for ingress only once – when the data is originally received. All other data transferred into Workers via the Cribl HTTP Source is not charged.

This use case is common in deployments where a customer-managed (on-prem) Worker or Edge node 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 Filesystem 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 Worker Group/Fleet in Cribl.Cloud.
- For an on-prem Worker/Edge Node configure a Filesystem Collector to send data to the Cribl HTTP Destination, and from there to the Cribl HTTP Source in Cribl.Cloud.

Configuration Requirements

The key points about configuring this architecture are:

- The Cribl HTTP Destination must be on a Worker Node that is connected to the same Leader as the Cribl HTTP Source.
- You must specify the same Leader Address on the Worker Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Worker Node's UI. Then select ⚙ **Settings** (lower left) > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Worker Nodes must be identical. In this example, both Worker Nodes must point to `cribl-leader`:

  ```
  distributed:
    mode: master
    master:
      host: cribl-leader
      port: 4200
  ```

- When you configure the Cribl HTTP Destination, its **Cribl endpoint** field must point to the **Address** and **Port** you've configured on the Cribl HTTP Source.
- Finally, it's important to understand the special way the Cribl HTTP Source handles internal fields.

Configuring the Cribl HTTP Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl HTTP**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Cribl HTTP**. Next, click **+ Add New** to open a **Cribl HTTP** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Cribl HTTP Source definition.

Address: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).

Port: Enter the port number to listen on, e.g., `10200`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled Defaults to `No`. When toggled to `Yes`, exposes this section's remaining fields.

Certificate name: Select a predefined certificate from the drop-down. A **Create** button is available to create a new certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`, exposes these two additional fields:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.
- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

Enable Persistent Queue defaults to `No`. When toggled to `Yes`:

Mode: Choose a mode from the drop-down:

- With `Smart` mode, PQ will write events to the filesystem only when it detects backpressure from the processing engine.
- With `Always On` mode, PQ will always write events directly to the queue before forwarding them to the processing engine.

Max buffer size: Maximum number of events to hold in-memory before dumping them to disk.

Commit frequency: Number of events to send before committing that Stream has read them.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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.)

;

7.9.4. Cribl TCP

The internal Cribl TCP 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.

> Type: **System and Internal** | TLS Support: **YES** | Event Breaker Support: **No**

You might choose this Source over the Cribl HTTP Source in certain circumstances, such as when a firewall or proxy allows raw TCP egress. In single-instance mode or for testing, you can substitute the TCP JSON Source. (However, this substitution will not provide the single-billing benefits described in the next section.)

How It Works

You can use the Cribl TCP Source to send data between Workers. In a hybrid Cloud deployment, using this Source ensures that you're billed for ingress only once – when the data is originally received. All other data transferred into Workers via the Cribl TCP Source is not charged.

This use case is common in deployments where a customer-managed (on-prem) Worker or Edge node 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 Filesystem 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 Worker Group/Fleet in Cribl.Cloud.
- For an on-prem Worker/Edge Node configure a Filesystem Collector to send data to the Cribl TCP Destination, and from there to the Cribl TCP Source in Cribl.Cloud.

Configuration Requirements

The key points about configuring this architecture are:

- The Cribl TCP Destination must be on a Worker Node that is connected to the same Leader as the Cribl TCP Source.
- You must specify the same Leader Address on the Worker Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Worker Node's UI. Then select ⚙ **Settings** (lower left) > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Worker Nodes must be identical. In this example, both Worker Nodes must point to `cribl-leader`:

  ```
  distributed:
    mode: master
    master:
      host: cribl-leader
      port: 4200
  ```

- When you configure the Cribl TCP Destination, its **Cribl endpoint** field must point to the **Address** and **Port** you've configured on the Cribl TCP Source.
- Finally, it's important to understand the special way the Cribl TCP Source handles internal fields.

Configuring the Cribl TCP Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **Cribl TCP**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Cribl TCP**. Next, click **+ Add New** to open a **Cribl TCP** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Cribl TCP Source definition.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on, e.g., `10300`.

Additional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

Enable Persistent Queue defaults to `No`. When toggled to `Yes`:

Mode: Choose a mode from the drop-down:

- With `Smart` mode, PQ will write events to the filesystem only when it detects backpressure from the processing engine.
- With `Always On` mode, PQ will always write events directly to the queue before forwarding them to the processing engine.

Max buffer size: Maximum number of events to hold in memory before dumping them to disk. Defaults to `1000`.

Commit frequency: Number of events to send before committing that Cribl 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

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.

;

7.9.5. Cribl Stream (Deprecated)

> This Source is deprecated as of Cribl Stream 3.5. Please instead use the Cribl TCP or the Cribl HTTP Source instead 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: **System and Internal** | TLS Support: **YES** | Event Breaker Support: **No**

Use New Sources Instead

The replacement Sources, Cribl TCP and the Cribl HTTP, don't rely on IP filtering, for the following reasons:

- Load balancers and/or proxies between the Cribl Destination and Cribl Source can change the IP address, resulting in a bad match and rejected ingest.
- A Lookup table of all IP addresses needed to be sent to each Worker Node/Edge Node from the Leader, which is not scalable.
- The Lookup table of IP addresses required constant communication between the Worker Node/Edge Nodes and the Leader, making this fragile and placing an arbitrary reliance on the Leader that shouldn't be there.

How It Works

You can use the Cribl Stream Source to send data between Workers. In a hybrid Cloud deployment, using this Source ensures that you're billed for ingress only once – when the data is originally received. All other data transferred into Workers via the Cribl Stream Source is not charged.

This use case is common in deployments where a customer-managed (on-prem) Worker sends data to a Worker in Cribl Cloud, for additional processing and then routing to Destinations.

As one usage example, assume that you want to send data from one Worker Node/Edge Node deployed on-prem, to another that is deployed in Cribl Cloud. You could do the following:

- Create an on-prem Filesystem 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 Filesystem 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 New** 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 **+ Add New** to open a **Cribl Stream** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Cribl Stream Source definition.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Use this default option to enter the shared secret that clients must provide in the `authToken` header field. Exposes an **Auth token** field for this purpose. (If left blank, unauthenticated access will be permitted.) A **Generate** link is available if you need a new secret.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. The secret can reside in Cribl Stream's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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`

;

7.10. System

7.10.1. Exec

The Exec Source enables you to periodically execute a command and collect its `stdout` output. This is typically used in cases when Cribl Stream cannot accomplish collection with native Collectors or other Sources.

> Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **Yes**

> This Source allows you to run **almost anything** on the host system. Make sure you understand the security and other impacts of commands you plan to execute, before proceeding.

Especially for monitoring or polling, you'll need to receive the command or script's output periodically. For this reason, the Exec Source lets you specify when the command or script should run, by time interval or by cron-style schedule.

Here are a few examples of what you can run from an Exec Source:

- Database queries.
- Custom commands to poll databases or apps.
- `ping` to check latency.
- `ps -ax` to get a list of running processes.
- `SNMP GET` requests to query an SNMP agent about network entities.
- `netstat -natp` to get lists of sockets and TCP ports, and what they're doing.
- `mtr -c 1 --report --json 8.8.8.8` to run a traceroute on a location (here, `8.8.8.8`), and packaging up a report in JSON format.

Configuring an Exec Source

In the **QuickConnect** UI: Click **+ New Source**, or click **+ Add** beside **Sources**. From the resulting drawer's tiles, select [**System and Internal** >] **Exec**. Next, click **Select Existing**.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **Exec**.

General Settings

Enabled: Defaults to `Yes`.

Input ID: Enter a unique name to identify this Exec Source definition.

Command: Command to execute; supports Bourne shell syntax.

Schedule type: Use the buttons to select either `Interval` or `Cron`. Customize the behavior in the corresponding field below the button.

- **Interval**: Specify how often, in seconds, the command should run. Defaults to `60`.
- **Schedule**: Enter a cron expression. (You enter the cron expression in UTC time, but the resulting **Estimated Schedule** displays in local time.)

Optional Settings

Max retries: Maximum number of retry attempts in the event that the command fails.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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.

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.

;

7.10.2. File Monitor

The File Monitor Source collects log (text) files, and generates events from lines or records in those files. This Source is available in both Stream and Edge. On Cribl.Cloud, it is available for all Edge Fleets, and for all Stream Worker Groups **except** for `default`.

> Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **Yes**

Discovering and Filtering Files to Monitor

To produce its initial list of files to monitor, the File Monitor Source runs a discovery procedure at a configurable **Polling interval**. The Source then applies a **Filename allowlist** to filter the initial list down into its final form.

For any file that makes it past the allowlist, the File Monitor Source keeps track of how far into the file it has read. For a given polling interval, this enables the Source to ignore inactive files, while watching active ones (including newly created files). The state information that the Source maintains for all monitored files appears in the **Status** tab, as explained in the Examples. When files are renamed (due to log rotation, for example), the Source handles this gracefully, as explained in Monitoring Renamed Files' State.

How does the File Monitor Source discover files in the first place? You have the choice of two **Discovery mode**s:

In **Auto** mode, the Source automatically discovers files that running processes have open for writing.

- This option's usefulness is limited to situations, such as demonstrations or testing, where you need to discover all the files that it would be possible to monitor.
- Use with caution if the extra costs of "grabbing everything" are a concern.

In **Manual** mode, you have fine-grained control over what to monitor and what to ignore. In this mode, the Source can perform either or both of two actions:

- Discovering the files within a directory, and to a depth, that you specify.
- Monitoring a set of files where each needs to be handled in a unique way. Here, you'll typically create a set of Sources, each of which monitors one file. See Monitoring Individual Files.

Configuring a File Monitor Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **File Monitor**. Next, click **Add New**.

Or, in the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**. From the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **File Monitor**.

General Settings

Enabled: Defaults to `Yes`.

Input ID: Enter a unique name to identify this File Monitor Source definition.

Discovery mode: Use the buttons to select one of these options.

- **Auto**: Tells the Source to automatically discover files that running processes have open for writing.

- **Manual**: Tells the Source to discover the files within the **Search path** that you specify, down to the **Max depth**.

  In **Search path**, if the files you want to match all reside within one directory (and/or its subdirectories), you should enter that path, along with a **Filename allowlist** (as described in Optional Settings).

  By default, the **Max depth** field is empty. This means that 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.

  If you want the Source to monitor one file per Source, you should simply enter that file's complete path. This makes the **Filename allowlist** irrelevant, and you should leave the **Max depth** empty.

Both modes allow you to set the **Polling interval**, which otherwise defaults to 10 seconds.

Optional Settings

Filename allowlist: This field supports wildcard syntax, and supports the exclamation mark (`!`) for negation. 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`.

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.

> The **Filename allowlist** is really a blocklist combined with an allowlist. All the negation patterns should be first – if you add a negation pattern, drag it towards the beginning with the other negations. As long as the negation patterns are together, followed by all the match ("allow") patterns, the order of the individual patterns within each group does not matter.

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.

> This Source defaults to QuickConnect.

Examples

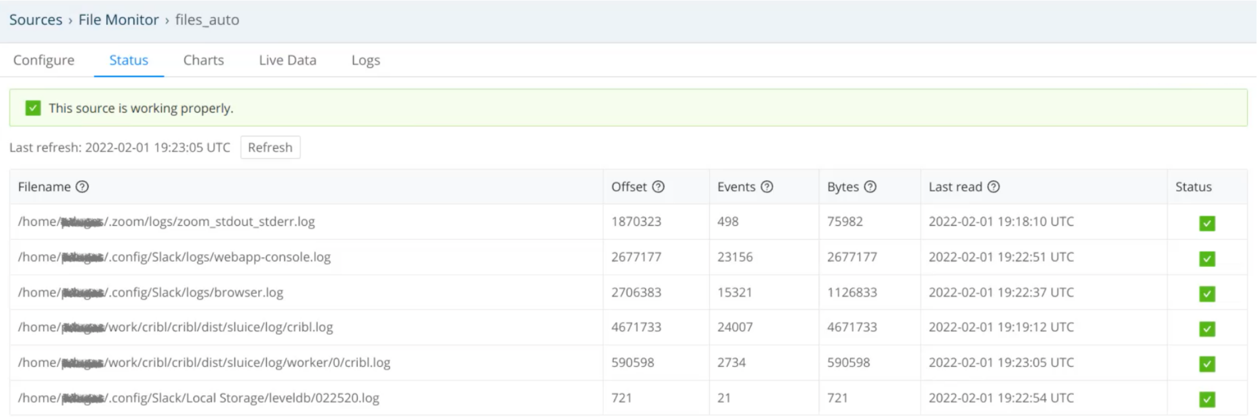
Following the fine points in these examples, along with Monitoring Individual Files, will enable you to monitor precisely what you want to (as opposed to monitoring too few or too many files).

Monitoring a Subset of Log Files in a Directory

Suppose you added a File Monitor Source on a Linux machine, set your **Discovery mode** to **Manual**, and specified your home directory as the **Search path** (e.g., `/home/bogart/`). You're using the Chrome browser.

You do not want to monitor Cribl Stream's own log files, nor any log files generated by Chrome. To exclude them, you add `!*cribl*access.log` and `!*chrome*` to your **Filename allowlist**. You **do** want to monitor any other log files that the Source can discover, so you also add `*log`.

After a while, the **Status** tab looks like this:



☑ This source is working properly.

Last refresh: 2022-02-01 19:23:05 UTC Refresh

Filename ⓘ	Offset ⓘ	Events ⓘ	Bytes ⓘ	Last read ⓘ	Status
/home/[redacted]/.zoom/logs/zoom_stdout_stderr.log	1870323	498	75982	2022-02-01 19:18:10 UTC	✓
/home/[redacted]/.config/Slack/logs/webapp-console.log	2677177	23156	2677177	2022-02-01 19:22:51 UTC	✓
/home/[redacted]/.config/Slack/logs/browser.log	2706383	15321	1126833	2022-02-01 19:22:37 UTC	✓
/home/[redacted]/work/cribl/cribl/dist/sluice/log/cribl.log	4671733	24007	4671733	2022-02-01 19:19:12 UTC	✓
/home/[redacted]/work/cribl/cribl/dist/sluice/log/worker/0/cribl.log	590598	2734	590598	2022-02-01 19:23:05 UTC	✓
/home/[redacted]/.config/Slack/Local Storage/leveldb/022520.log	721	21	721	2022-02-01 19:22:54 UTC	✓

The Status tab

Teleporting with Cribl Edge to Discover Files to Monitor

Suppose you want to explore the files on a particular host, in order to decide what to monitor. You can teleport into an Edge Node, select the desired host, and view the **Files** tab to see what files are being written there.

Based on what you observe, you can then decide how to monitor the files that interest you. You'll set your **Discovery mode** to **Manual** in order to appropriately configure file paths and allowlists, whether you use a single Source or a set of multiple Sources. In the latter case, see Monitoring Individual Files.

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 the last-read point. So 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.

Monitoring Individual Files

Why monitor a set of individual files, as opposed to whatever files might exist in a specified directory? Your purposes could include any or all of the following:

- Sending data from each individual file to a different Destination (or, from some files within the set to different Destinations).
- Sending data from each individual file to a Pack that's appropriate for that file type.
- Enriching the file's data with metadata required by the receiver. Here, metadata means fields such as `index` and `sourcetype`, and receivers that might require it include Splunk and Elasticsearch.

Source per File

In this kind of scenario, you'll configure a separate Source for each file you want to monitor, in the following way:

First, as the **Discovery mode**, select **Manual**.

Next, in the **Search path** field, specify the full path of the file you want to monitor. The **Filename allowlist** will have no effect, and you should leave **Max depth** empty.

Then, choose one of the following options to complete the process:

- Use QuickConnect to configure Pipelines and Destinations for each Source.
- Set pre-processing Pipelines, or configure specific Packs, for each Source.
- Use **Processing Settings** > **Fields** to set metadata for the file you want to monitor.

Cribl has found that this Source-per-file approach yields better performance, granularity, and supportability than alternative approaches.

Monitoring Both Directories and Individual Files

Suppose you need to configure both a set of Sources, each monitoring an individual file, **and** another Source monitoring multiple files in the same directory path.

This works, but it requires the allowlist for the multiple-file-monitoring Source to exclude each of the files that's been assigned its own Source. Otherwise, you risk seeing data from the same file show up both Sources (which is redundant and can cause double-billing).

To sum up: Keep careful track of all the file paths and directory/wildcard patterns you're monitoring. See Optional Settings for details about working with allowlists.

;

7.10.3. 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.

> Type: **System and Internal** | TLS Support: **N/A** | Event Breaker Support: **No**
>
> Cribl Edge Workers support System Metrics only when running on Linux, not on Windows.

Configuring Cribl Stream to Collect System Metrics

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**System and Internal** >] **System Metrics**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**System and Internal** >] **System Metrics**. Next, click **+ Add New** to open a **System Metrics** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Optional Settings

Polling interval: How often to collect metrics, in seconds. Defaults to `10`.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Host Metrics

Use the buttons to select a level of detail.

- **Basic** enables minimal metrics, averaged or aggregated.
- **All** enables full, detailed metrics, specified for individual CPUs, interfaces, and so on.
- **Custom** displays sub-menus and buttons from which you can choose a level of detail (**Basic**, **All**, **Custom**, or **Disabled**) for each type of event.
- **Disabled** means that no metrics will be generated.

The meaning of **All** and **Disabled** are self-evident. **Basic** and **Custom** have different meanings depending on event type, as follows:

System

Basic level captures load averages, uptime, and CPU count.

Custom toggles **Process** metrics on or off; these are metrics for the numbers of processes in various states.

CPU

Basic level captures active, user, system, idle, iowait percentages over all CPUs.

Custom toggles the following on or off: **Per CPU metrics**, **Detailed metrics** (meaning, for all CPU states), and **CPU time metrics** (meaning raw, monotonic CPU time counters).

Memory

Basic level captures captures total, used, available, `swap_free`, and `swap_total`.

Custom toggles **Detailed metrics** on or off. Detailed means for all memory states.

Network

Basic level captures bytes, packets, errors, connections over all interfaces.

Custom exposes the following:

- The **Interface filter**, which specifies which network interfaces to include or exclude (all are included if the filter is empty).
- **Per interface metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Protocol metrics** toggle appears, allowing you to choose whether to generate metrics for ICMP, ICMPMsg, IP, TCP, UDP, and UDPLite.

Disk

Basic level captures disk used in percent, bytes read and written, and read and write operations, over all mounted disks.

Custom exposes the following:

- The **Device filter**, which specifies which block devices to include or exclude (all are included if the filter is empty). Wildcards and `!` (not) operators are supported.
- The **Mountpoint filter**, which specifies which filesystem mountpoints to include or exclude (all are included if the filter is empty). Wildcards and `!` (not) operators are supported.
- The **Filesystem type filter**, which specifies which filesystem types to include or exclude (all are included if the filter is empty). Wildcards and `!` (not) operators are supported.
- **Per device metrics**, which toggle on or off.
- **Detailed metrics**, which toggle on or off. If on, the **Enable inode metrics** toggle appears, allowing you to choose whether to generate metrics for filesystem inodes.

Container Metrics

Use the buttons to select a level of detail.

- **Basic** generates the server event and per running container events.
- **All** adds per-device and detailed metrics.
- **Custom** provides controls for customizing which containers to generate metrics from.
- **Disabled** means that no metrics will be generated.

The **Custom** buttons displays additional controls, as follows:

- **Container Filters**: Enter a filter expression to govern which containers to generate metrics from. Leave empty (the default) to generate metrics from all containers.
- **All containers**: Toggle to `Yes` to include stopped and paused containers.
- **Per device metrics**: Toggle to `Yes` to generate separate metrics for each device.
- **Detailed metrics**: Toggle to `Yes` to generate full container metrics.

The **Basic**, **All**, and **Custom** buttons provide the following **Advanced Settings**, which are different from those in the main **Advanced Settings** tab:

- **Docker socket**: Enter the full path(s) for Docker's UNIX domain socket. Defaults to `/var/run/docker.sock` and `/run/docker.sock`.
- **Docker timeout**: Timeout, in seconds, for the Docker API. Defaults to `5`.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

If desired, choose a **Pipeline** from the drop-down if you want to process data from this Source before sending it through the Routes.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Enable disk persistence: Whether to save metrics to disk. Defaults to `No`. When toggled to `Yes`, exposes this section's remaining fields.

Bucket time span: What time range of events to hold in each bucket. Default value: `10m` (10 minutes).

Max data size: Maximum disk space the persistent metrics can consume. Once reached, Cribl 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`.

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.

;

7.11. 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.

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **YES**

Configuring Cribl Stream to Receive AppScope Data

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **AppScope**. Next, click either **+ Add New** 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**. Or, from the top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **AppScope**. Next, click **+ Add New** to open an **AppScope** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this AppScope Source definition.

UNIX domain socket: When toggled to `Yes`, exposes the following two fields to specify a file-backed UNIX domain socket connection to listen on.

- **UNIX socket path**: Path to the UNIX domain socket. Defaults to `$CRIBL_HOME/state/appscope.sock`.
- **UNIX socket permissions**: Permissions to set for this socket, e.g., `777`. If empty, Cribl Stream will use the runtime user's default permissions.

When **UNIX domain socket** is set to `No` (the default), you instead see the following two fields to specify a network host and port.

- **Address**: Enter the hostname/IP on which to listen for AppScope data. (E.g., `localhost`.) Defaults to `0.0.0.0`, meaning all addresses.
- **Port**: Enter the port number to listen on.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Use this default option to enter the shared secret clients must provide in the `authToken` header field. Click **Generate** if you need a new auth token. If empty, unauthenticated access will be permitted.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token described above. The secret can reside in Cribl Stream's internal secrets manager or (if enabled) in an external KMS. Click **Create** if you need to configure a new secret.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

This left tab is displayed only when the Optional Settings > **UNIX domain socket** slider is set to `No`. It provides the following options.

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

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.

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.

Internal Settings

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`

Examples

Cribl Cloud – TLS

- An `in_appscope` TLS Source is preconfigured for you in Cribl Cloud, using port `10090`.

- Send it AppScope data using this command:

```
./scope run -c in.logstream.<tenant-ID>.cribl.cloud:10090 -- ps -ef
```

Cribl Cloud – TCP

- Add a new AppScope Source and assign it **Port**: `10091`.

- Send it AppScope data using this command:

```
./scope run -c tcp://in.logstream.<tenant-ID>.cribl.cloud:10091 \
>     -- curl -so /dev/null https://wttr.in/94105
```

;

7.12. CrowdStrike

Cribl Stream supports receiving data from the CrowdStrike Falcon 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 Falcon 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**

Configuring a CrowdStrike Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Pull** >] **CrowdStrike**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Pull** >] **CrowdStrike**. Next, click **+ Add New** to open a **CrowdStrike** > **New Source** modal that 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 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 for filtering and grouping in Cribl Stream. 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. 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

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.

Endpoint: The S3 service endpoint you want CrowdStrike to use. If empty, Cribl Stream will automatically construct the endpoint from the Region.

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: 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.

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.

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.

;

7.13. 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`.

Configuring Cribl Stream to Ingest Datadog Agent Output

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Datadog Agent**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Datadog Agent**. Next, click **+ Add New** to open a **Datadog Agent** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Datadog Agent Source definition.

Address: Enter hostname/IP to listen for Datadog Agent data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.
- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

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.

Environment: If you're using GitOps, 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 active requests: Maximum number of active requests per worker process. Use `0` for unlimited.

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.

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.

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`. Cannot be sent to Cribl Stream even when you **are** sending other types there.	`DD_APM_DD_URL`

Managing API Keys

Suppose your data flow runs from Datadog Agents, to Cribl 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.

;

7.14. 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)

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Elasticsearch API**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Elasticsearch API**. Next, click **+ Add New** to open an **Elasticsearch API** > **New Source** modal that provides the following options and fields.

> 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.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Authentication

In the **Authentication type** drop-down, select one of the following options:

- **None**: Don't use authentication.

- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials. Click **Generate** if you need a new password.

- **Basic (credentials secret)**: Provide username and password credentials referenced by a secret. Select a stored text secret in the resulting **Credentials secret** drop-down, or click **Create** to configure a new secret.

- **Auth tokens**: Use HTTP token authentication. Click **+ Add Token** and, in the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header. Click **Generate** if you need a new token. Click **+ Add Token** to display additional rows to specify more tokens.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

  > Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle this to `Yes` to add request headers to events, in the `__headers` field.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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 as the default for backward compatibility.
- `8.3.2` – 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`.

Remove headers: Enter any headers that you want removed from proxied requests. Press `Tab` or `Return` to separate header names.

Proxy request timeout: How long, in seconds, to wait for a proxy request to complete before aborting it. Defaults to `60` seconds; minimum timeout is `1` second.

To understand how **Proxy request timeout** interacts with the `X-Forwarded-For` header, see Overriding `__srcIpPort` with Client IP/Port.

Authentication method: Select one of the following options.

- **None**: Don't use authentication.
- **Manual**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Secret**: This option exposes a **Credentials secret** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source’s data to one or more Destinations via independent, direct connections.

Field Normalization

The Elasticsearch API input normalizes the following fields:

- `@timestamp` becomes `_time` at millisecond resolution.
- `host` is set to `host.name`.
- Original object `host` is stored in `__host`.

The Elasticsearch Destination does the reverse, and it also recognizes the presence of `__host`.

Internal Settings

Cribl 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`
- `__srcIpPort` – See details below.
- `__type`

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the Elasticsearch client sending data to this Source.

When any proxies (including load balancers) lie between the Elasticsearch client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** is set to `Yes`, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Configuring Elastic Beats

Beats are open-source data shippers that act as agents, sending data to Elasticsearch (or to other services, in this case Cribl 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.

In Cribl Stream 3.5.2 and later, the **Extra HTTP Headers** and **API Version** settings facilitate working with Beats. If you're on a pre-3.5.2 version of Cribl Stream, see alternate configuration procedures in the next section.

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 Elastic Beats with Pre-3.5.2 Cribl Stream

> These instructions configure current Beats versions (v.8.x or newer) with legacy Cribl Stream versions. Some earlier **Beats** versions require different configuration. For specific questions, please contact us on Cribl's Community Slack.

To an Elastic Beat sending data to this Source, the Source will appear as an instance of Elasticsearch. If you are on Cribl Stream 3.5.1 or older, you must configure the Source as follows to successfully interact with an Elastic Beat:

1. Set `output.elasticsearch.allow_older_versions` to `true` to prevent errors of the form: `Elasticsearch is too old. Please upgrade the instance.` In general, `Elasticsearch` errors in the Filebeat logs will actually be about Cribl Stream.

2. Disable the index lifecycle management (ILM) feature by setting `setup.ilm.enabled` to `false`. If ILM is enabled, the Beat will probably be unable to send data to Cribl Stream. In this case, you will see corresponding errors or warnings in the Beat's log.

To make these configuration changes, adapt the appropriate snippet below, and add it to the Beat config file:

filebeat.yml (on-prem)   filebeat.yml (Cloud)

```
setup.ilm.enabled: false
output.elasticsearch:
  hosts: ["http://<instance_FQDN>:9200"]
  allow_older_versions: true
```

;

7.15. 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)

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **HTTP**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **HTTP**. Next, click **+ Add New** to open a **HTTP** > **New Source** modal that provides the following options and fields.

> 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.

Address: Enter the hostname/IP on which to listen for HTTP(S) data. (E.g., `localhost` or `0.0.0.0`.)

Port: Enter the port number.

Authentication Settings

Auth tokens: Shared secrets to be provided by any client (`Authorization: <token>`). Click **Generate** to create a new secret. If empty, unauthenticated access **will be permitted**.

Optional Settings

Cribl HTTP event API: Base path on which to listen for Cribl HTTP API requests. To construct the actual endpoint, Cribl 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. Maximum payload size is 2MB.

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 for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable proxy protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. This setting affects how the Source handles the `__srcIpPort` field.

Capture request headers: Toggle this to `Yes` to add request headers to events, in the `__headers` field.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Activity log sample rate: Determines how often request activity is logged at the `info` level. The default `100` value logs every 100th value; a `1` value would log every request; a `10` value would log every 10th request; etc.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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":"mySource", "fieldA":"valueA", "fieldB":"valueB"}
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": "
{\"message\":\"Something informative happened\", \"severity\":\"INFO\"}"}
```

Note 1: Events can be sent as separate POSTs, but Cribl **highly** recommends combining multiple events in newline-delimited groups, and POSTing them together.

Note 2: If an HTTP(S) source is routed to a Splunk destination, fields within the JSON payload are mapped to Splunk fields. Fields that do not have corresponding (native) Splunk fields become index-time fields. For example, let's assume we have a HTTP(S) event like this:

```
{"_time":1541280341, "host":"myHost", "source":"mySource", "_raw":"this is a sample
event ", "fieldA":"valueA"}
```

Here, `_time`, `host` and `source` become their corresponding fields in Splunk. The value of `_raw` becomes the actual body of the event, and `fieldA` becomes an index-time field. (`fieldA`::`valueA`).

Examples

Cribl 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
'{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

Cribl Endpoint – Multiple Events

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d
$'{"_raw":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"} \n {"_raw":"this is another sample event ",
"host":"myOtherHost", "source":"myOtherSource", "fieldA":"valueA",
"fieldB":"valueB"}'
```

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
myToken42' -d '{"event":"this is a sample event ", "host":"myHost",
"source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'

curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myToken42'
-d '{"event":"this is a sample event ", "host":"myHost", "source":"mySource",
"fieldA":"valueA", "fieldB":"valueB"}'
```

> For Splunk HEC, the token specification can be either `Splunk <token>` or `<token>`.

Cribl Cloud – Single Event

1. Generate and copy a token in your Cribl Cloud instance's HTTP Source > **General Settings**.

2. From the command line, use `https`, your Cribl.Cloud portal's **Ingest Endpoint** and port, and the token's value:

```
curl -k https://in.logstream.<tenant-ID>.cribl.cloud:10080/cribl/_bulk -H
'Authorization: <token_value>' -d '{"_raw":"this is a sample event ",
"host":"myHost", "source":"mySource", "fieldA":"valueA", "fieldB":"valueB"}'
```

;

7.16. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Raw HTTP**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Raw HTTP**. Next, click **+ Add New** to open a **Raw HTTP** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Raw HTTP Source definition.

Address: Enter the address to bind on. Defaults to `0.0.0.0` (all addresses).

Port: Enter the port number to listen on.

Authentication Settings

Auth tokens: Shared secrets to be provided by any client. Click **Generate** to create a new secret. If empty, permits open access.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

Processing Settings

Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maxiumum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2. This setting affects how the Source handles the `__srcIpPort` field.

Allowed URI paths: List of URI paths accepted by this input. Supports wildcards, e.g., `/api/v*/hook`. Defaults to `*`, which allows all paths.

Allowed HTTP methods: List of HTTP methods accepted by this input. Supports wildcards, e.g., `P*, GET`. Defaults to `*`, which allows all methods.

Max active requests: Maximum number of active requests allowed for this Source, per Worker Process. Defaults to `256`. Enter `0` for unlimited.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Request timeout (seconds): How long to wait for an incoming request to complete before aborting it. The default `0` value means wait indefinitely.

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.)

;

7.17. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Metrics**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Metrics**. Next, click **+ Add New** to open a **Metrics** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

UDP port: Enter the UDP port number to listen on. Not required if listening on TCP.

TCP port: Enter the TCP port number to listen on. Not required if listening on UDP.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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 `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

IP allowlist regex: Regex matching IP addresses that are allowed to send data. Defaults to `.*` (i.e., all IPs.)

Max buffer size (events) : Maximum number of events to buffer when downstream is blocking. Defaults to `1000`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source's data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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
dim3 - 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.

;

7.18. OpenTelemetry (OTel)

Cribl Stream supports receiving trace and metric events from OTLP-compliant senders. (Cribl plans to add support for log events once logs support in the OpenTelemetry protocol graduates from experimental status.)

> Type: **Push** | TLS Support: **YES** | Event Breaker Support: **No**

Supported and Unsupported Input Data

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**.

Don't Compress Inbound Data

This Source does not support currently compressed inbound data. Therefore, in your OTEL Exporter, pass the `compression: none` parameter as shown in this example:

```
exporters:
  otlp:
    endpoint: "https://<Cribl_IP_address>:4317"
    compression: none
    tls:
      insecure: false
      insecure_skip_verify: true
```

Configuring an OTel Source

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **OpenTelemetry**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **OpenTelemetry**. Next, click **+ Add New** to open a **OpenTelemetry** > **New Source** modal that 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 Source.

General Settings

Input ID: Unique ID for this Source. E.g., `OTel042`.

Address: Enter the hostname/IP to listen to. Defaults to `0.0.0.0`.

Port: By default, OTel applications send output to port `4317`. Do not change this setting unless the OTel application whose data you want Cribl Stream to collect is using a different port.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

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. 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`.

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.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.

- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

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.

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.

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.pem
   -days 420
   ```

This example command will generate both a self-signed cert named `myCert.pem` (certified for 420 days), and an unencrypted, 2048-bit RSA private key named `myKey.pem`.

2. Configure the **TLS Settings (Server Side)**. Toggle **Enabled** to `Yes`, then:

   - Enter the appropriate values in the **Certificate name**, **Private key path**, and **Certificate path** fields. A **Create** link is available if you need a new certificate, and **Certificate name** also works as a drop-down to allow you to choose from any existing certificates.
   - Leave the **CA certificate path** field empty.
   - Leave **Authenticate client (mutual auth)** toggled to `No`.

3. Configure the OTel client. See the OTel Collector TLS Configuration Settings README for an explanation of the relevant settings. The config file might be named `otel-config.yaml`, `otel-local-config.yaml`, or just `config.yaml`, depending on your environment. This YAML file will have an `exporters` section, which you must edit to include an `otlp` sub-section, as follows:

   - Add an `endpoint` whose value is the IP address of either (a) the Cribl 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:

```yaml
exporters:
  otlp:
    endpoint: "https://<Cribl_IP_address>:4317"
    tls:
      insecure: false
      insecure_skip_verify: true
```

;

7.19. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **SNMP Trap**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **SNMP Trap**. Next, click **+ Add New** to open a **SNMP Trap** > **New Source** modal that provides the following options and fields.

> Cribl Stream ships with an SNMP Trap Source preconfigured to listen on Port 9162. You can clone or directly modify this Source to further configure it, and then enable it.

General Settings

Input ID: Enter a unique name to identify this Source definition.

Address: Address to bind on. Defaults to `0.0.0.0` (all addresses).

UDP Port: Port on which to receive SNMP traps. Defaults to `162`.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Processing Settings

Fields

In this section, you can add Fields to each event using Eval-like functionality.

Name: Field name.

Value: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

IP allowlist regex: Regex matching IP addresses that are allowed to send data. Defaults to `.*`, i.e., all IPs.

Max buffer size (events) : Maximum number of events to buffer when downstream is blocking. Defaults to `1000`.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source’s data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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` : In this particular Source, this field uses a pipe (|) symbol to separate the source IP address and the port, in this format: `event.__srcIpPort = ${rInfo.address}|${rInfo.port};`

- `__snmpVersion`: Acceptable values are `0`, `2`, or `3`. These respectively indicate SNMP v1, v2c, and v3.
- `__snmpRaw`: Buffer containing Raw SNMP packet

Considerations for Working with SNMP Trap Data

- It's possible to work with SNMP metadata (i.e., we'll decode the packet). Options include dropping, routing, etc.

- SNMP packets can be forwarded to other SNMP destinations. However, the contents of the incoming packet **cannot** be modified – i.e., we'll forward the packets verbatim as they came in.

- SNMP packets can be forwarded to non-SNMP destinations (e.g., Splunk, Syslog, S3, etc.).

- Non-SNMP input data **cannot** be sent to SNMP destinations.

;

7.20. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Syslog**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Syslog**. Next, click **+ Add New** to open a **Syslog** > **New Source** modal that provides the following options and fields.

> 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.

Address: Enter the hostname/IP on which to listen for data., E.g. `localhost` or `0.0.0.0`.

UDP port: Enter the UDP port number to listen on. Not required if listening on TCP.

> The maximum supported inbound UDP message size is 16,384 bytes.

TCP port: Enter the TCP port number to listen on. Not required if listening on UDP.

Optional Settings

Fields to keep: List of fields from source data to retain and pass through. Supports wildcards. Defaults to `*` wildcard, meaning keep all fields. Fields **not** specified here (by wildcard or specific name) will be removed from the event.

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. 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 `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

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: Enable this if messages are prefixed with a byte length, according to RFC 5425 or RFC 6587.

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.)

Default timezone: Timezone to assign to timestamps that omit timezone info. Accept the default `Local` value, or use the drop-down list to select a specific timezone by city name or GMT/UTC offset.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source’s data to one or more Destinations via independent, direct connections.

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`
- `__syslogFail`: `true` for data that fails RFC 3164/5424 validation as syslog format.

;

7.21. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **TCP JSON**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **TCP JSON**. Next, click **+ Add New** to open a **TCP JSON** > **New Source** modal that provides the following options and fields.

> 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.

Address: Enter hostname/IP to listen for TCP JSON data. E.g., `localhost` or `0.0.0.0`.

Port: Enter the port number to listen on.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Use this default option to enter the shared secret that clients must provide in the `authToken` header field. Exposes an **Auth token** field for this purpose. (If left blank, unauthenticated access will be permitted.) A **Generate** link is available if you need a new secret.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. The secret can reside in Cribl Stream's internal secrets manager or (if enabled) in an external KMS. A **Create** link is available if you need a new secret.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

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":"valueA", "fieldB":"valueB"}
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "
{\"message\":\"Something informative happened\", \"severity\":\"INFO\"}"}
```

TCP JSON Field Mapping to Splunk

If a TCP JSON Source is routed to a Splunk destination, fields within the JSON payload are mapped to Splunk fields. Fields that do not have corresponding (native) Splunk fields become index-time fields. For example, let's assume we have a TCP JSON event as below:

```
{"_time":1541280341, "host":"myHost", "source":"mySource", "_raw":"this is a sample
event ", "fieldA":"valueA"}
```

Here, `_time`, `host`, and `source` become their corresponding fields in Splunk. The value of `_raw` becomes the actual body of the event, and `fieldA` becomes an index-time field (`fieldA`::`valueA``).

Examples

Testing TCP JSON In

This first example simply tests that data is flowing in through the Source:

1. Configure Cribl 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 `<tenant-ID>` placeholder with the tenant 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: `in.logstream.<tenant-ID>.cribl.cloud` – you can simply copy/paste your Cribl Cloud portal's **Ingest Endpoint** here

Port: `10070`

TLS Settings (Client Side)

Enabled: `Yes`

Validate server certs: `Yes`

;

7.22. 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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **TCP** . Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **TCP**. Next, click **+ Add New** to open a **TCP** > **New Source** modal that provides the following options and fields.

> 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.

Address: Enter hostname/IP to listen for raw TCP data. E.g., `localhost` or `0.0.0.0`.

Port: Enter port number.

Enable Header: Toggle to `Yes` to indicate that client will pass a header record with every new connection. The header can contain an `authToken`, and an object with a list of fields and values to add to every event. These fields can be used to simplify Event Breaker selection, routing, etc. Header format: `{ "authToken" : "myToken", "fields": { "field1": "value1", "field2": "value2" }}`.

- **Shared secret (authToken)**: Shared secret to be provided by any client (in `authToken` header field). Click **Generate** to create a new secret. If empty, unauthenticated access will be permitted.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Server Side)

Enabled defaults to `No`. When toggled to `Yes`:

Certificate name: Name of the predefined certificate.

Private key path: Server path containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`.

Passphrase: Passphrase to use to decrypt private key.

Certificate path: Server path containing certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

CA certificate path: Server path containing CA certificates (in PEM format) to use. Path can reference `$ENV_VARS`.

Authenticate client (mutual auth): Require clients to present their certificates. Used to perform mutual authentication using SSL certs. Defaults to `No`. When toggled to `Yes`:

- **Validate client certs**: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

- **Common name**: Regex matching subject common names in peer certificates allowed to connect. Defaults to `.*`. Matches on the substring after `CN=`. As needed, escape regex tokens to match literal characters. E.g., to match the subject `CN=worker.cribl.local`, you would enter: `worker\.cribl\.local`.

Minimum TLS version: Optionally, select the minimum TLS version to accept from connections.

Maximum TLS version: Optionally, select the maximum TLS version to accept from connections.

Persistent Queue Settings

In this section, you can optionally specify persistent queue storage, using the following controls. This will buffer and preserve incoming events when a downstream Destination is down, or exhibiting backpressure.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Processing Settings

Custom Command

In this section, you can pass the data from this input to an external command for processing before the data continues downstream.

Enabled: Defaults to `No`. When toggled to `Yes`:

Command: Enter the command that will consume the data (via `stdin`) and will process its output (via `stdout`).

Arguments: Click **+ Add Argument** to add each argument for the command. You can drag arguments vertically to resequence them.

Event Breakers

Event Breaker rulesets: A list of event breaking rulesets that will be applied to the input data stream before the data is sent through the Routes. Defaults to `System Default Rule`.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum `10` ms, default `10000` (10 sec), maxiumum `43200000` (12 hours).

Fields

In this section, you can add Fields to each event using Eval-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Enable Proxy Protocol: Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol v1 or v2.

IP allowlist regex: Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e,. all IPs).

Max active connections: Maximum number of active connections allowed per Worker Process. Defaults to `1000`. Set a lower value if connection storms are causing the Source to hang. Set `0` for unlimited connections.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Connected Destinations

Select **Send to Routes** to enable conditional routing, filtering, and cloning of this Source's data via the Routing table.

Select **QuickConnect** to send this Source’s data to one or more Destinations via independent, direct connections.

Internal Fields

Cribl 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`:

```
cat test.raw | nc --ssl --ssl-verify in.logstream.<tenant-ID>.cribl.cloud 10060
```

;

7.23. 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**

Currently, this Source supports only client certificate (mutual TLS) authentication. Cribl plans to add Kerberos authentication support in a future release.

Upstream Prerequisites

This page assumes that:

- You already have Windows Event Forwarding set up;
- Pointed to one or more Windows Event Collectors (WECs);
- Using client certificate authentication over HTTPS.

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

In the **QuickConnect** UI: Click **+ New Source** or **+ Add Source**. From the resulting drawer's tiles, select [**Push** >] **Windows Event Forwarder**. Next, click either **+ Add New** 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 top nav of a Cribl Edge instance or Fleet, select **More** > **Sources**.

From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Windows Event Forwarder**. Next, click **+ Add New** to open a **Windows Event Forwarder** > **New Source** modal that provides the following options and fields.

General Settings

Input ID: Enter a unique name to identify this Windows Event Forwarder Source definition.

Address: Enter the hostname/IP on which to listen for Windows events data. (E.g., `localhost` or `0.0.0.0`.)

Port: Enter the port number. The default, 5986, is the port used by Windows Event Collector for HTTPS-based subscriptions.

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`.

> 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.

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.

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping in Cribl Stream. Use a tab or hard return between (arbitrary) tag names.

Subscriptions

Click on **+ 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, and 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 only events subsequent to the subscription, depending on the setting of **Read existing events**.

Compression: Sets whether Windows clients compress the events sent to Cribl Stream, using the Streaming Lossless Data Compression (SLDC) algorithm. Defaults to `Yes`.

Query builder mode: See the explanation in Configuring Queries below.

Queries: See the explanation in Configuring Queries below.

Targets: Set the DNS names of the clients that should receive this subscription. Wildcard-matching is supported.

Configuring Queries

Queries determine which events to send from the clients. **At least one query is required.** The format is derived from the XPath implementation used by Windows Event Collector.

You have two mode options for providing queries: **Simple** or **Raw XML**.

Select **Simple** if you need to manually build queries. Click **+ Add query** to define a 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.

Select **Raw XML** if you already have an XPath query.

Example

When creating a subscription in a Windows Event Collector, you can view the generated XML/XPath query. Consider a subscription that returns all events from the Security log that are of severities Critical, Error, or Warning, and that occurred within the last 24 hours. This subscription would generate XML like this:

```
<QueryList>
  <Query Id="0" Path="Security">
    <Select Path="Security">*[System[(Level=1  or Level=2 or Level=3) and
TimeCreated[timediff(@SystemTime) &lt;= 86400000]]]</Select>
  </Query>
</QueryList>
```

To use this subscription in a Cribl 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.

Enable Persistent Queue: Defaults to `No`. When toggled to `Yes`:

Mode: Select a condition for engaging persistent queues.

- `Smart`: This default option will engage PQ only when the Source detects backpressure from the Cribl Stream data processing engine.
- `Always On`: This option will always write events into the persistent queue, before forwarding them to the Cribl Stream data processing engine.

Max buffer size: The maximum number of events to hold in memory before reporting backpressure to the Source. Defaults to `1000`.

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

Max file size: The maximum data volume to store in each queue file before closing it and (optionally) applying the configured **Compression**. Enter a numeral with units of KB, MB, etc. If not specified, Cribl 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, Cribl Stream will stop queueing data, and will apply the **Queue-full behavior**. Enter a numeral with units of KB, MB, etc. If not specified, the implicit `0` default will enable Cribl Stream to fill all available disk space on the volume.

Queue file path: The location for the persistent queue files. Defaults to `$CRIBL_HOME/state/queues`. To this field's specified path, Cribl 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.

> Setting the PQ **Mode** to `Always On` can degrade throughput performance. Select this mode only if you want guaranteed data durability. As a trade-off, you might need to either accept slower throughput, or provision more machines/faster disks.

Processing Settings

Pre-Processing

In this section's **Pipeline** drop-down list, you can select a single existing Pipeline to process data from this input before the data is sent through the Routes.

Advanced Settings

Allow MachineID mismatch: Set this to `Yes` if you do not want to verify that the events sent by a client match the client's certificate Common Name (Subject CN). If set to `No`, events where the `MachineID` (by default the client's machine name, like `CLIENT1.domainName.com`) do not match the CN will be rejected.

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.

CA fingerprint override: If the top–level Intermediate Authority (IA) certificate in a certificate chain does not match the **first** certificate in the chain, enter the IA cert's SHA1 fingerprint here. (By default, Cribl Stream calculates and uses the fingerprint of the first cert found in the CA cert file. If the IA directly signing the client and server certs is in a different position, this overrides that default to prevent a mismatch.)

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`
- `__srcIpPort` – See details below.
- `__subscriptionName`
- `__subscriptionVersion`
- `__headers` – Added only when **Advanced Settings** > **Capture request headers** is set to `Yes`.

Overriding `__srcIpPort` with Client IP/Port

The `__srcIpPort` field's value contains the IP address and (optionally) port of the WEF client sending data to this Source.

When any proxies (including load balancers) lie between the WEF client and the Source, the last proxy adds an `X-Forwarded-For` header whose value is the IP/port of the original client. With multiple proxies, this header's value will be an array, whose first item is the original client IP/port.

If `X-Forwarded-For` is present, and **Advanced Settings** > **Enable proxy protocol** is set to `No`, the original client IP/port in this header will override the value of `__srcIpPort`.

If **Enable proxy protocol** is set to `Yes`, the `X-Forwarded-For` header's contents will **not** override the `__srcIpPort` value. (Here, the upstream proxy can convey the client IP/port without using this header.)

Configuring Upstream Clients/Senders

This section summarizes how to configure Windows endpoints/senders to forward events to this Cribl 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. 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>`

   - 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 `https`. (This Source does not currently support `http` with Kerberos.)

   - 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:

- 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 enable the Proxy Protocol setting in the **Advanced** section of the Source configuration. Ensure that the correct outbound firewall port is opened on the client.

- 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.

- Check Cribl Stream for any errors, as well as the `EventForwarding-Plugin` and `Windows Remote Management` event logs on the clients.

Troubleshooting

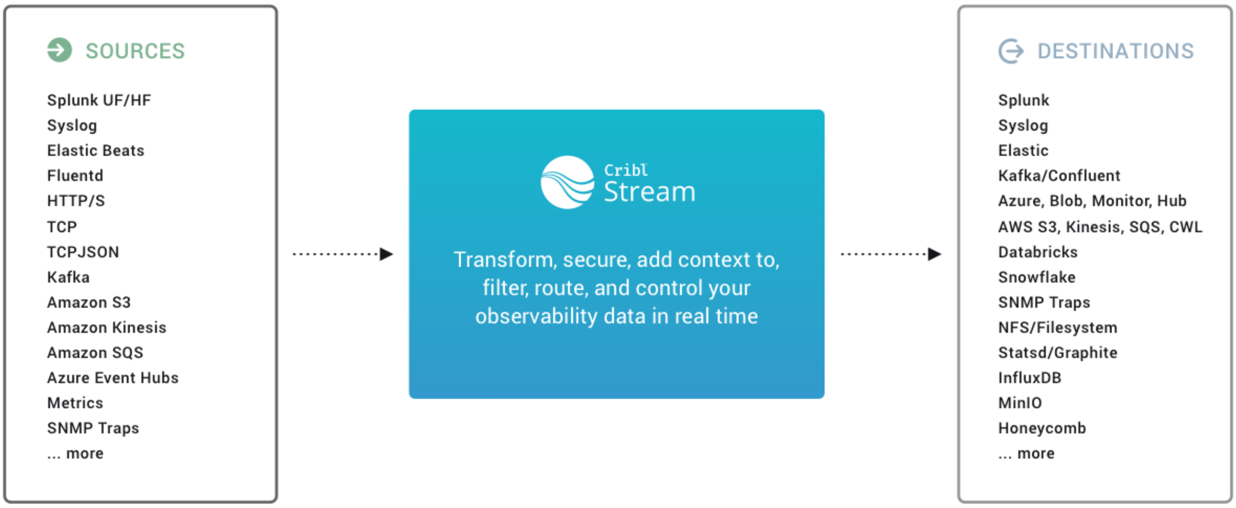
When using auto-enroll, the Eventlog-ForwardingPlugin Operational logs might display an error of this form:
`If Kerberos mechanism is used, verify that the client computer and the destination computer are joined to a domain.`

To resolve this: In your auto-enroll template's **Client-Server Authentication Properties**, make sure the **Subject Name format** is set to **Common name**.

;

8. Destinations

Cribl Stream can send data to various Destinations, including Splunk, Kafka, Kinesis, InfluxDB, Snowflake, Databricks, TCP JSON, and many others.



Streaming Destinations

Destinations that accept events in real time are referred to as streaming Destinations:

- Splunk Single Instance
- Splunk Load Balanced
- Splunk HEC
- Amazon Kinesis Streams
- Amazon CloudWatch Logs
- Amazon SQS
- Azure Monitor Logs
- Azure Event Hubs
- Google Chronicle
- Google Cloud Pub/Sub
- StatsD
- StatsD Extended
- Graphite
- TCP JSON

- Syslog
- Kafka
- Confluent
- Cribl HTTP
- Cribl TCP
- Elasticsearch
- Honeycomb
- New Relic Logs & Metrics
- New Relic Events
- SNMP Trap
- InfluxDB
- Wavefront
- SignalFx
- Sumo Logic
- Datadog
- Webhook
- Prometheus
- Grafana Cloud
- Loki
- OpenTelemetry (OTel)
- DataSet
- Humio HEC
- Cribl Stream

Non-Streaming Destinations

Destinations that accept events in groups or batches are referred to as non-streaming Destinations:

- Amazon S3 Compatible Stores
- Azure Blob Storage
- Google Cloud Storage
- Filesystem/NFS
- MinIO

> 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.

Internal and Peer-to-Peer Destinations

These special-purpose Destinations route data within your Cribl Stream deployment, or among Workers across distributed or hybrid Cloud deployments:

- Default: Specify a default output from among your configured Destinations.
- Output Router: A "meta-Destination." Configure rules that route data to multiple configured Destinations.
- DevNull: Simply drops events. Preconfigured and active when you install Cribl Stream, so it requires no configuration. Useful for testing.
- Cribl HTTP: Send data among peer Worker Nodes over HTTP.
- Cribl TCP: Send data among peer Worker Nodes over TCP.
- Cribl Stream (Deprecated): Use either Cribl HTTP or Cribl TCP instead.
- **SpaceOut**: This experimental Destination is undocumented. Be careful!

How Does Non-Streaming Delivery Work

Cribl 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).

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.)

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.

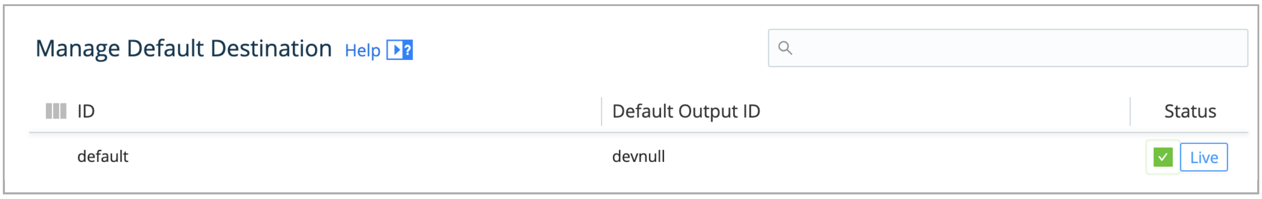
Configuring Destinations

For each Destination **type**, you can create multiple definitions, depending on your requirements.

To configure Destinations, select **Data** > **Destinations** from Cribl Stream's global top nav (single-instance deployments), or from a Worker Group's/Fleet's top nav (distributed deployments). On the resulting **Data Destinations** page's tiles or left menu, select the desired type, then click **+ Add New**.

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.

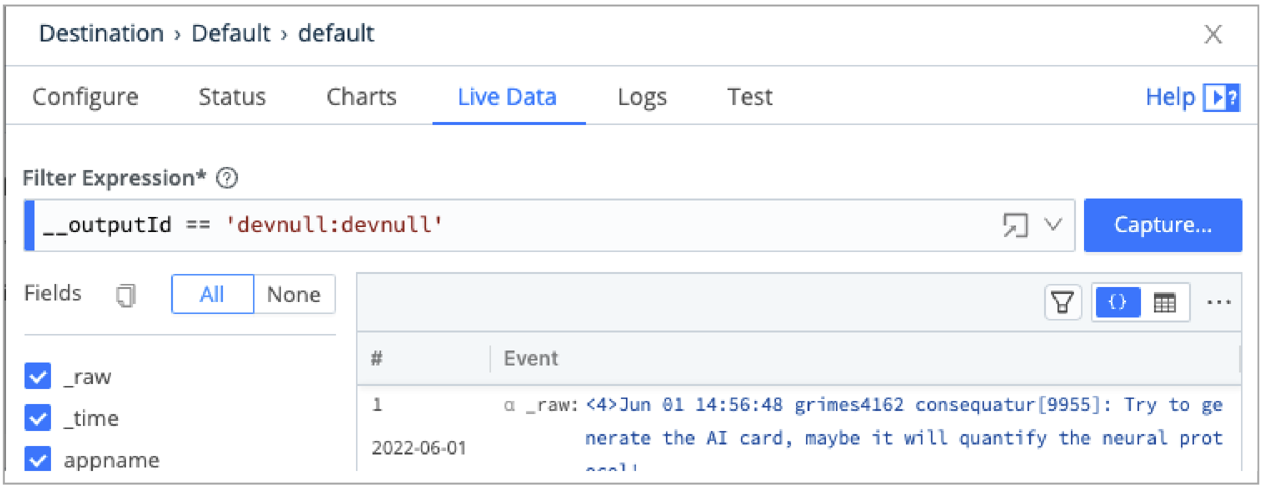


Manage Default Destination Help

ID	Default Output ID	Status
default	devnull	✓ 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.



Filter Expression*

```
__outputId == 'devnull:devnull'
```

☑ _raw
☑ _time
☑ appname

```
1           α _raw: <4>Jun 01 14:56:48 grimes4162 consequatur[9955]: Try to ge
                    nerate the AI card, maybe it will quantify the neural prot
2022-06-01          ocol!
```

Destination modal > Live Data tab

;

8.1. Amazon

8.1.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **CloudWatch Logs**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Amazon** > **CloudWatch Logs**.
Next, click **+ Add New** to open an **Amazon CloudWatch Logs** > **New Destination** modal that provides the following options and fields.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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.)

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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.

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

Endpoint: CloudWatch Logs service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, use this field to point to a CloudWatchLogs-compatible endpoint.

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.

;

8.1.2. 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

Configuring Cribl Stream to Output to Amazon Kinesis Data Streams

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **Kinesis**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Amazon** > **Kinesis**. Next, click **+ Add New** to open an **Amazon Kinesis** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Kinesis definition.

Stream name: Enter the name of the Kinesis Data Stream to which to send events.

Region: Select the AWS Region where the Kinesis Data Stream is located.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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 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.)

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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.

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

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`.

Put request concurrency: Maximum number of ongoing put requests before blocking. Defaults to `5`.

Max record size (KB, uncompressed): Maximum size of each individual record before compression. For non-compressible data, 1MB (the default) is the maximum recommended size.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum.

Reuse connections: Whether to reuse connections between requests. The default setting (`Yes`) can improve performance.

Reject unauthorized certificates: Whether to accept certificates that cannot be verified against a valid Certificate Authority (e.g., self-signed certificates). Defaults to `Yes` .

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Format

Currently, outputted events use the following record format:

- Header line containing information about the payload (currently supports one type, as shown below).
- Newline-Delimited JSON (that is, each Kinesis record will contain multiple events, in **ndjson** format).

Record payloads (including header and body) will be gzip-compressed, and then Kinesis will base64-encode them.

Sample Kinesis Record

```
{"format":"ndjson","count":8,"size":3960}
{"_raw":"07-03-2018 18:33:51.136 -0700 ERROR TcpOutputFd - Read error. Connection
reset by peer","_meta":"timestartpos::0 timeendpos::29 _subsecond::.136
date_second::51 date_hour::18 date_minute::33 date_year::2018 date_month::july
date_mday::3 date_wday::tuesday date_zone::-420 punct::--_::._-___-
__.____","_time":"1530668031","source":"/mnt-
big/ledion/hwf/var/log/foo/food.log","host":"ledion-
hwf","sourcetype":"food","index":"_internal","cribl_pipe":"foo2"}
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to
127.0.0.1:10000 closed. Read error. Connection reset by
peer","_meta":"timestartpos::0 timeendpos::29 _subsecond::.136 date_second::51
date_hour::18 date_minute::33 date_year::2018 date_month::july date_mday::3
date_wday::tuesday date_zone::-420 punct::--_::._-____-
___...:_.__.____","_time":"1530668031","source":"/mnt-
big/ledion/hwf/var/log/foo/food.log","host":"ledion-
hwf","sourcetype":"food","index":"_internal","cribl_pipe":"foo2"}
...
```

;

8.1.3. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **S3**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Amazon** > **S3**. Next, click **+ Add New** to open an **Amazon S3** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this S3 definition.

S3 bucket name: Name of the destination S3 Bucket. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. E.g., referencing a Global Variable: `myBucket-${C.vars.myVar}`.

> Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization. If you want to use event-level variables in file paths, Cribl recommends specifying them in the **Partitioning Expression** field (described below), because this is evaluated for each file.

Staging location: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance. (This field is not displayed or available on Cribl.Cloud-managed 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 `none`. Cribl recommends setting this to `gzip`. This setting is not available when **Data format** is set to `Parquet`.

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to ``` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ```, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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.

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_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.

Parquet Settings

To write out Parquet files, note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.
- The S3 Collector currently does not support ingesting data in the Parquet format. Therefore, data that you export in Parquet format cannot be replayed.

Parquet schema: Select a schema from the drop-down. The default `sample_parquet` schema is always available.

> Cribl recommends that you add a new schema – or clone the sample schema and modify it to suit your needs – via **Processing** > **Knowledge** > **Parquet Schemas**. Schemas that you add there will become available in this drop-down. For details, see Parquet Schemas.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

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.

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 slider'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, the endpoint will be automatically constructed from the region.

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.

Amazon S3 Permissions

The following permissions are needed to write to an Amazon S3 bucket:

- `s3:ListBucket`
- `s3:GetBucketLocation`
- `s3:PutObject`

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

;

8.1.4. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Amazon** > **SQS**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Amazon** > **SQS**. Next, click **+ Add New** to open an **Amazon SQS** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this SQS Destination.

Queue name: The name, URL, or ARN of the SQS queue to send events to. This value must be a JavaScript expression (which can evaluate to a constant), enclosed in single quotes, double quotes, or backticks.
To specify a non-AWS URL, use the format: `'{url}/<queueName>'`. (E.g., `':port/<myQueueName>'`.)

Queue type: The queue type used (or created). Defaults to `Standard`. `FIFO` (First In, First Out) is the other option.

Optional Settings

Message group ID: This parameter applies only to queues of type FIFO. Enter the tag that specifies that a message belongs to a specific message group. (Messages belonging to the same message group are processed in FIFO order.) Defaults to `cribl`. Use event field `__messageGroupId` to override this value.

Create queue: Specifies whether to create the queue if it does not exist. Defaults to `Yes`.

Region: Region where SQS queue is located.

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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.)

Authentication

Use the **Authentication Method** buttons to select an AWS authentication method.

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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

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.

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

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`

;

8.2. Azure

8.2.1. Azure Blob Storage

Azure Blob Storage is a non-streaming Destination type. Cribl Stream does **not** have to run on Azure in order to deliver data to it. Azure Data Lake Storage Gen2 (hierarchical namespace) is also supported.

> Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Configuring Cribl Stream to Output to Azure Blob Storage

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Blob Storage**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Azure** > **Blob Storage**. Next, click **+ Add New** to open an **Azure Blob Storage** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Container name: Enter the container name. (A container organizes a set of blobs, similar to a directory in a file system.)

> Container names can include only lowercase letters, numbers, and/or hyphens (-). This restriction is imposed by Azure.

Blob prefix: Root directory to prepend to path before uploading.

Staging location: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance. (This field is not displayed or available on Cribl.Cloud-managed 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** buttons to select one of these options:

- **Manual**: Use this default option to enter your Azure Storage connection string directly. Exposes a **Connection string** field for this purpose. (If left blank, Cribl 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=<your-account-key>`

A fictitious example, using Microsoft's recommended HTTPS option, is: `DefaultEndpointsProtocol=https;AccountName=storagesample;AccountKey=12345678...32`

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 `none`. Cribl recommends setting this to `gzip`. 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: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl 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.

Parquet Settings

To write out Parquet files, note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Parquet schema: Select a schema from the drop-down. The default `sample_parquet` schema is always available.

> Cribl recommends that you add a new schema – or clone the sample schema and modify it to suit your needs – via **Processing** > **Knowledge** > **Parquet Schemas**. Schemas that you add there will become available in this drop-down. For details, see Parquet Schemas.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files reaching this size will be closed and moved to the final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Default: `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Default: `100`.

> Cribl 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: 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 slider'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`

;

8.2.2. 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.

Configuring Cribl Stream to Output to Azure Event Hubs

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Event Hubs**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Azure** > **Event Hubs**. Next, click **+ Add New** to open an **Azure Event Hubs** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Azure Event Hubs definition.

Brokers: List of Event Hub Kafka brokers to connect to. (E.g., `yourdomain.servicebus.windows.net:9093`.) Find the hostname in Shared Access Policies, in the host portion of the primary or secondary connection string.

Event Hub name: The name of the Event Hub (a.k.a., Kafka Topic) on which to publish events. Can be overwritten using the `__topicOut` field.

Optional Settings

Acknowledgments: Control the number of required acknowledgments. Defaults to `Leader`.

Record data format: Format to use to serialize events before writing to the Event Hub Kafka brokers. Defaults to `JSON`.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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)

Enabled Defaults to `Yes`.

Validate server certs: Defaults to `No` – and for Event Hubs, this must always be disabled.

Authentication

Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `Yes`. (Toggling to `No` hides the remaining settings in this group.)

SASL mechanism: SASL (Simple Authentication and Security Layer) authentication mechanism to use, `PLAIN` is the only mechanism currently supported for Event Hub Kafka brokers.

Username: The username for authentication. For Event Hub, this should always be `$ConnectionString`.

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials. The password is your Event Hubs primary or secondary connection string.
  From Microsoft's documentation, the format is:
  `Endpoint=sb://<FQDN>/;SharedAccessKeyName=<KeyName>;SharedAccessKey=<KeyValue>`

  Example entry:
  `Endpoint=sb://dummynamespace.servicebus.windows.net/;SharedAccessKeyName=dummyaccesskeyname;SharedAccessKey=5dOntTRytoC24opYThisAsit3is2B+OGY1US/fuL3ly=`

- **Secret**: This option exposes a **Password (text 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.

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

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.

Environment: If you're using GitOps, 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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

;

8.2.3. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Azure** > **Monitor Logs**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Azure** > **Monitor Logs**. Next, click **+ Add New** to open an **Azure Monitor Logs** > **New Destination** modal that provides the following options and fields.

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`.

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

Resource ID: Resource ID of the Azure resource to associate the data with. This populates the `_ResourceId` property, and allows the data to be included in resource-centric queries. (Optional, but if this field is not specified, the data will not be included in resource-centric queries.)

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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_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

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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096`.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low settings could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.

- If keepalives are not supported by the server (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname, Cribl 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.

;

8.3. Google Cloud

8.3.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Chronicle**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Google Cloud** > **Chronicle**. Next, click **+ Add New** to open a **Google Cloud Chronicle** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Chronicle output definition.

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.

Optional Settings

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

The Google Chronicle API key is required to complete this part of the Destination definition.

Use the **Authentication Method** buttons to select one of these options:

- **Manual**: In the resulting **API key** field, enter your Google Chronicle API key.
- **Secret**: This option exposes a **Secret** drop-down, in which you can select a stored secret that references your Google Chronicle API key. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of `KB`, `MB`, etc. Defaults to 1 MB.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of `KB`, `MB`, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block**

option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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).

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: Toggle to `Yes` if you want Cribl Stream to compress the payload body before sending.

Request timeout: Enter an amount of time, in seconds, to wait for a request to complete before aborting it.

Request concurrency: Enter the maximum number of ongoing requests to allow before blocking.

Max body size (KB): Enter a maximum size, in KB, for the request body.

Max events per request: Enter the maximum number of events to include in the request body. Defaults to `0` (unlimited).

Flush period (sec): Enter the maximum time to allow between requests. Be aware that small values could cause the payload size to be smaller than the configured **Max body size**.

Extra HTTP Headers: Click **+ Add Header** to insert extra headers as **Name/Value** pairs.

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.

;

8.3.2. 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, the following access permissions must be set on the Cloud Storage side:

- Fine-grained access control must be enabled on the buckets.
- The Google service account or user must have the Storage Admin or Owner role.

For details, see the Cloud Storage Overview of Access Control and Understanding Roles documentation.

Configuring Cribl Stream to Output to Cloud Storage Destinations

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Cloud Storage**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Google Cloud** > **Cloud Storage**. Next, click **+ Add New** to open a **Google Cloud > Cloud Storage** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Cloud Storage definition.

Bucket name: Name of the destination bucket. This value can be a constant. or a JavaScript expression that can be evaluated only at init time. E.g., referencing a Global Variable: `myBucket-${C.vars.myVar}`.

Region: Region where the bucket is located.

Staging location: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance. (This field is not displayed or available on Cribl.Cloud-managed 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 `none`. Cribl recommends setting this to `gzip`. This setting is not available when **Data format** is set to `Parquet`.

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

- **Manual**: With this default option, authentication is via HMAC (Hash-based Message Authentication Code). To create a key and secret, see Google Cloud's Managing HMAC Keys for Service Accounts documentation. This option exposes these two fields:
    - **Access key**: Enter the HMAC access key.
    - **Secret key**: Enter the HMAC secret.
- **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_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.

Parquet Settings

To write out Parquet files, note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Parquet schema: Select a schema from the drop-down. The default `sample_parquet` schema is always available.

> Cribl recommends that you add a new schema – or clone the sample schema and modify it to suit your needs – via **Processing** > **Knowledge** > **Parquet Schemas**. Schemas that you add there will become available in this drop-down. For details, see Parquet Schemas.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

> Cribl 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.

;

8.3.3. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Google Cloud** > **Pub/Sub**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Google Cloud** > **Pub/Sub**. Next, click **+ Add New** to open a **Google Cloud Pub/Sub** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Pub/Sub output definition.

Topic ID: ID of the Pub/Sub topic to send events to.

Optional Settings

Create topic: Toggle to `Yes` if you want Cribl 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication Method** buttons to select one of these options:

- **Auto**: This option uses the environment variables `PUBSUB_PROJECT` and `PUBSUB_CREDENTIALS`, and requires no configuration here.
- **Manual**: This default option displays a **Service account credentials** field for you to enter the contents of your service account credentials file (a set of JSON keys), as downloaded from Google Cloud. To insert the file itself, click the upload button at this field's upper right. As an alternative, you can use environment variables, as outlined here.
- **Secret**: This option exposes a drop-down in which you can select a stored secret that references the service account credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This 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`; `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_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

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.

Let's Change the Topic

The Pub/Sub Destination supports alternate topics specified at the event level in the `__topicOut` field. So (e.g.) if a Pub/Sub Destination is configured to send to main topic `topic1`, and Cribl 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.

;

8.4. Kafka

8.4.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Kafka**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Kafka**. Next, click **+ Add New** to open a **Kafka** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Kafka definition.

Brokers: List of Kafka brokers to connect to. (E.g., `localhost:9092`.)

Topic: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.

Optional Settings

Acknowledgments: Select the number of required acknowledgments. Defaults to `Leader`.

Record data format: Format to use to serialize events before writing to Kafka. Defaults to `JSON`.

Compression: Codec to compress the data before sending to Kafka. Select `None`, `Gzip` `Snappy`, or `LZ4`.

> Cribl strongly recommends enabling compression. Doing so improves Cribl 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down.**Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

TLS Settings (Client Side)

Enabled 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).

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

Minimum TLS version: Optionally, select the minimum TLS version to use when connecting.

Maximum TLS version: Optionally, select the maximum TLS version to use when connecting.

Certificate name: The name of the predefined certificate.

CA certificate path: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

Private key path (mutual auth): Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Certificate path (mutual auth): Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.

Passphrase: Passphrase to use to decrypt private key.

Authentication

This section governs SASL (Simple Authentication and Security Layer) authentication to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

SASL mechanism: Use this drop-down to select the SASL authentication mechanism to use. The mechanism you select determines the controls displayed below.

PLAIN, SCRAM-256, or SCRAM-512

With any of these authentication mechanisms, select one of the following buttons:

Manual: Displays **Username** and **Password** fields to enter your Kafka credentials directly.

Secret: This option exposes a **Credentials secret** drop-down in which you can select a stored text secret that references your Kafka credentials. A **Create** link is available to store a new, reusable secret.

GSSAPI/Kerberos

Selecting Kerberos as the authentication mechanism displays the following options:

Keytab location: Enter the location of the key table file for the authentication principal.

Principal: Enter the authentication principal, e.g.: `kafka_user@example.com`.

Broker service class: Enter the Kerberos service class for Kafka brokers, e.g.: `kafka`.

Schema Registry

This section governs Kafka Schema Registry Authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (E.g., `http://<hostname>:8081`.)

Default key schema ID: Used when `__keySchemaIdOut` is not present to transform key values. Leave blank if key transformation is not required by default.

Default value schema ID: Used when `__valueSchemaIdOut` not present to transform `_raw`. Leave blank if value transformation is not required by default.

TLS enabled: defaults to `No`. When toggled to `Yes,` displays the following TLS settings for the Schema Registry (in the same format as the TLS Settings (Client Side) above):

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to **No**.
- **Server name (SNI)**: Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.
- **Minimum TLS version**: Optionally, select the minimum TLS version to use when connecting.
- **Maximum TLS version**: Optionally, select the maximum TLS version to use when connecting.
- **Certificate name**: The name of the predefined certificate.
- **CA certificate path**: Path on client containing CA certificates (in PEM format) to use to verify the server's cert. Path can reference `$ENV_VARS`.

- **Private key path (mutual auth)**: Path on client containing the private key (in PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Certificate path (mutual auth)**: Path on client containing certificates in (PEM format) to use. Path can reference `$ENV_VARS`. **Use only if mutual auth is required**.
- **Passphrase**: Passphrase to use to decrypt private key.

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl 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

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.

Environment: If you're using GitOps, 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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

;

8.4.2. 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

In the **QuickConnect** UI: Click **+ Add** beside **Destinations**. From the resulting drawer's tiles, select **Confluent Cloud**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Confluent Cloud**. Next, click **+ Add New** to open a **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Brokers: List of Confluent Cloud brokers to connect to. (E.g., `myAccount.confluent.cloud:9092`.)

Topic: The topic on which to publish events. Can be overwritten using event's `__topicOut` field.

Optional Settings

Acknowledgments: Select the number of required acknowledgments. Defaults to `Leader`.

Record data format: Format to use to serialize events before writing to Kafka. Defaults to `JSON`.

Compression: Codec to use to compress the data before sending to Kafka. Select `None`, `Gzip`, or `Snappy`.

Backpressure behavior: Select whether to block, drop, or queue incoming events when all receivers are exerting backpressure. Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`, `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 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)

Enabled When toggled to `Yes` (the default):

Autofill?: This setting is experimental.

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

> With a dedicated Confluent Cloud cluster hosted in Microsoft Azure, be sure to specify the **Server name (SNI)**. If this is omitted, Confluent Cloud might reset the connection to Cribl 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.

Authentication

Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.

Enabled: Defaults to `No`. When toggled to `Yes`:

- **SASL mechanism**: Select the SASL (Simple Authentication and Security Layer) authentication mechanism to use. Defaults to `PLAIN`. `SCRAM-SHA-256` and `SCRAM-SHA-512` are also available.

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.

- **Secret**: This option exposes a **Credentials secret** drop-down, in which you can select a stored secret that references the credentials described above. A **Create** link is available to store a new, reusable secret.

Schema Registry

This section governs Kafka Schema Registry Authentication for Avro-encoded data with a schema stored in the Confluent Schema Registry.

Enabled: Defaults to `No`. When toggled to `Yes`, displays the following controls:

Schema registry URL: URL for access to the Confluent Schema Registry. (E.g., `http://localhost:8081`.)

Default key schema ID: Used when `__keySchemaIdOut` is not present to transform key values. Leave blank if key transformation is not required by default.

Default value schema ID: Used when `__valueSchemaIdOut` not present to transform `_raw`. Leave blank if value transformation is not required by default.

TLS enabled: defaults to `No`. When toggled to `Yes,` displays the following TLS settings for the Schema Registry (in the same format as the TLS Settings (Client Side) above):

- **Validate server certs**: Require client to reject any connection that is not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to **No**.

- **Server name (SNI)**: Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

  > With a dedicated Confluent Cloud cluster hosted in Microsoft Azure, be sure to specify the **Server name (SNI)**. If this is omitted, Confluent Cloud might reset the connection to Cribl 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_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

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.

Environment: If you're using GitOps, 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:

- `__topicOut`
- `__key`
- `__headers`
- `__keySchemaIdOut`
- `__valueSchemaIdOut`

;

8.5. Metrics

8.5.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **Graphite**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Metrics** > **Graphite**. Next, click **+ Add New** to open a **Metrics > Graphite** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Graphite definition.

Destination protocol: Protocol to use when communicating with the Destination. Defaults to `UDP`.

Host: The hostname of the Destination.

Port: Destination port. Defaults to `8125`.

Optional Settings

Throttling: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.

Backpressure behavior: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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.)

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_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

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.

;

8.5.2. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **StatsD**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Metrics** > **StatsD**. Next, click **+ Add New** to open a **Metrics > StatsD** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this StatsD definition.

Destination protocol: Protocol to use when communicating with the Destination. Defaults to `UDP`.

Host: The hostname of the Destination.

Port: Destination port. Defaults to `8125`.

Optional Settings

Throttling: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.

Backpressure behavior: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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 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_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

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.

;

8.5.3. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Metrics** > **StatsD Extended**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Metrics** > **StatsD Extended**.
Next, click **+ Add New** to open a **Metrics > StatsD Extended** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this StatsD Extended definition.

Destination protocol: Protocol to use when communicating with the Destination. Defaults to `UDP`.

Host: The hostname of the Destination.

Port: Destination port. Defaults to `8125`.

Optional Settings

Throttling: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Rate (in bytes per second) at which at which to throttle while writing to an output. Also takes numerical values in multiples of bytes (KB, MB, GB, etc.). Default value of `0` indicates no throttling.

Backpressure behavior: Displayed only when **General Settings** > **Destination protocol** is set to `TCP`. Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed only when **General Settings** > **Destination protocol** is set to `TCP`, and only when **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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 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_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

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.

;

8.6. New Relic Ingest

8.6.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **New Relic Ingest** > **Events**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **New Relic Ingest** > **Events**. Next, click **+ Add New** to open a **New Relic Ingest > Events** > **New Destination** modal that provides the following options and fields.

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 uses 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 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

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. For the `Persistent Queue` option, see the section just below.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to `Persistent Queue`.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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 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_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

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 before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `1000` KB.

Max events per request: Maximum number of events to include in the request body. Defaults to `0`, allowing unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **+ Add Header** to insert extra headers as **Name**/**Value** pairs.

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.

;

8.6.2. 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
>
> As of Cribl Stream v.3.1.2, 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 as of v.3.1.2, 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **New Relic Ingest** > **Logs & Metrics**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **New Relic Ingest** > **Logs & Metrics**. Next, click **+ Add New** to open a **New Relic Ingest > Logs & Metrics** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this New Relic definition.

Region: Select which New Relic region endpoint to use.

Authentication Settings

> If an incoming event contains an internal field named `__newRelic_apiKey`, the New Relic Logs & Metrics Destination uses that field's value as the API key when sending the event to New Relic.
>
> For events that do not contain an `__newRelic_apiKey` field, the Destination uses whatever API key you have configured in the **Authentication method** settings.

Authentication method: Select one of the following buttons.

- **Manual**: This default option exposes an **API key** field. Directly enter your New Relic Ingest License API key, as you created or accessed it from New Relic's **account** drop-down. (For details, see the New Relic API Keys documentation.)

- **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references a New Relic Ingest License API key. A **Create** link is available to store a new, reusable secret.

Optional Settings

Log type: Name of the `logType` to send with events. E.g., `observability` or `access_log`.

> This sets a default. Where a `sourcetype` is specified in an event, it will override this value.

Log message field: Name of the field to send as the log `message` value. If not specified, the event will be serialized and sent as JSON.

Fields: Additional fields to (optionally) add, as **Name**-**Value** pairs. Click **+ Add Field** to add more.

- **Name**: Enter the field name.

- **Value**:JavaScript expression to compute field’s value, enclosed in single quotes, double quotes, or backticks. (Can evaluate to a constant.)

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. For the `Persistent Queue` option, see the section just below.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to `Persistent Queue`.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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_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

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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `1000` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **+ Add Header** to insert extra headers as **Name**/**Value** pairs.

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.

;

8.7. Prometheus

8.7.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Prometheus**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Prometheus**. Next, click **+ Add New** to open a **Prometheus** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Prometheus output definition.

Remote Write URL: The endpoint to send events to, e.g.: `http://localhost:9200/write`

Optional Settings

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.
- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.
- **Auth token (text secret)**: This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.
- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.
- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`; `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.

Advanced Settings

Validate server certs: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl 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. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

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

- 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 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname, Cribl 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.

;

8.7.2. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Grafana Cloud**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Grafana Cloud**. Next, click **+ Add New** to open a **Grafana Cloud** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Grafana Cloud output definition.

Loki URL: The endpoint to send log events to, e.g.: `https://logs-prod-us-central1.grafana.net`. This is the **Grafana Data Source settings** URL you wrote down earlier.

Prometheus URL: The endpoint to send metric events to, e.g.: `https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push`. This is the **Remote Write Endpoint** URL you wrote down earlier.

Optional Settings

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.

Advanced Settings

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned.

Compress: When the **Message format** is `JSON`, you can toggle this slider to `Yes` to GZIP-compress the data before sending to Grafana Cloud. (Applies only to Loki's JSON payloads. This slider is hidden when the **Message format** is `Protobuf`, because both Prometheus' and Loki's Protobuf implementations are Snappy-compressed by default.)

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

> Loki and Prometheus might complain about entries being delivered out of order when **Request concurrency** is set > `1` and any of **Flush period (sec)**, **Max body size (KB)**, or **Max events per request** are set to low values.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

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`.

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

- 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 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname, Cribl 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.

;

8.7.3. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Loki**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Loki**. Next, click **+ Add New** to open a **Loki** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Loki output definition.

Loki URL: The endpoint to send events to, e.g.: `https://logs-prod-us-central1.grafana.net`.

Optional Settings

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.

- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.

- **Auth token (text secret)**: This option exposes a drop-down in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.

- **Basic**: This default option displays fields for you to enter HTTP Basic authentication credentials. **Username** is the Loki **User**. **Password** is your API key in the Grafana Cloud domain.

- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.

Advanced Settings

Compress: When the **Message format** is `JSON`, you can toggle this slider to `Yes` to GZIP-compress the data before sending to Loki. (When the **Message format** is `Protobuf`, data is always Snappy-compressed, so this slider is hidden.)

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 CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `No`.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `1`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `15`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Message format: Whether to send events as `Protobuf` (the default) or `JSON`.

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.

;

8.8. Splunk

8.8.1. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Splunk** > **HEC**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Splunk** > **HEC**. Next, click **+ Add New** to open a **Splunk HEC** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Splunk HEC definition.

Load balancing: Set to `No` by default. When toggled to `Yes`, see Load Balancing Settings below.

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 **Optional Settings** > **Load balancing** is toggled to `Off`.

> For Splunk Cloud endpoints, change the default `http:` prefix to: `https:`.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays an **HEC Auth token** field for you to enter your Splunk HEC API access token.
- **Secret**: This option exposes an **HEC Auth token (text secret)** drop-down, in which you can select a stored secret that references the API access token described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

> For Splunk Cloud endpoints, change the **Splunk HEC endpoint**'s default `http:` prefix to: `https:`
>
> This Destination does not support Mutual TLS (mTLS).

Load Balancing Settings

Enabling the **Load balancing** slider displays the following controls:

Exclude Current Host IPs

This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

Splunk HEC Endpoints

The **Splunk HEC Endpoints** table is where you specify a known set of receivers on which to load-balance data. Click **+ Add Endpoint** to specify more receivers on new rows. Each row provides the following fields:

HEC Endpoint: Specify the URL to a Splunk HEC endpoint to send events to – e.g., `http://localhost:8088/services/collector/event`.

Load weight: Specify a weight to apply to the receiver for load-balancing purposes.

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.

See the explanation of how load balancing works below.

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`; `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_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

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: Toggle 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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`. Each request can potentially hit a different HEC receiver.

Max body size (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1`.

> - Retries happen on this flush interval.
> - Any HTTP response code in the `2xx` range is considered success.
> - Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
> - Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **+ Add Header** to add **Name**/**Value** pairs to pass as additional HTTP headers.

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.

How Does Load Balancing Work

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all HEC 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 all endpoints simultaneously, and the amount sent to each receiver depends on these parameters:

1. Respective destination **weight**.
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.

Example

Suppose we have two receivers, 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., receiver A will start the interval with **60** and receiver 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.

Notes on HTTP-based Outputs

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.
- If the server does not support keepalives – or if the server closes a pooled connection while idle – a new connection will be established for the next request.
- When resolving the Destination's hostname with load balancing disabled, Cribl Stream will pick the first IP in the list for use in the next connection. Enable Round-robin DNS to better balance distribution of events between Splunk HEC servers.
- See Splunk's documentation on editing `fields.conf` to ensure the visibility of index-time fields sent to Splunk by Cribl Stream.

;

8.8.2. Splunk Single Instance

The **Splunk Enterprise** 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Splunk** > **Single Instance**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Splunk** > **Single Instance**. Next, click **+ Add New** to open a **Splunk Single Instance** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Splunk Single Instance definition.

Address: Hostname of the Splunk receiver.

Port: The port number on the host.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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 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)

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 `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.

> ### 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_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

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.)

Minimize in-flight data loss: Directs Cribl Stream to check whether the indexer is shutting down, and if so, to stop sending data. This helps minimize data loss during shutdown. Toggle to `No` to disable this feature.

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.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Notes about Forwarding to Splunk

- Data sent to Splunk is not compressed.

- The only `ack` from indexers that Cribl 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.

;

8.8.3. 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.

How Does Load Balancing Work

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all receivers (listed as Destinations in the GUI). If FQDNs/hostnames are used as the Destination address and each resolves to, for example, 5 (unique) IPs, then each Worker Process will have its # of outbound connections = # of IPs x # of FQDNs for purposes of the SplunkLB output. Data is sent by all Worker Processes to all receivers simultaneously, and the amount sent to each receiver depends on these parameters:

1. Respective destination **weight**.
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.

Example

Suppose we have two receivers, 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., receiver A will start the interval with **60** and receiver 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.

Configuring Cribl Stream to Load-Balance to Multiple Splunk Destinations

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Splunk** > **Load Balanced**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Splunk** > **Load Balanced**. Next, click **+ Add New** to open a **Splunk Load Balanced** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Splunk LB Destination definition.

Toggling **Indexer discovery** to `Yes` enables automatic discovery of indexers in an indexer clustering environment. This hides both **Exclude current host IPs** and the **Destinations** section, and displays the following fields:

Site: Clustering site from which indexers need to be discovered. In the case of a single site cluster, `default` is the default entry.

Cluster Manager URI: Full URI of Splunk cluster manager, in the format: `scheme://host:port`. (Worker Nodes/Edge Nodes normally access the cluster manager on **port 8089** to get the list of currently online indexers.)

Refresh period: Time interval (in seconds) between two consecutive fetches of indexer list from cluster manager. Defaults to `300` seconds, i.e., 5 minutes.

Authentication method: Use the buttons to select one of these options for authenticating to cluster Manager for indexer discovery:

- **Manual**: In the resulting **Auth token** field, enter the required token.

- **Secret**: This option exposes a **Auth token (text secret)** drop-down, in which you can select a stored secret that references the auth token. A **Create** link is available to store a new, reusable secret.

> Each Worker Process performs its own indexer discovery according to the above settings.

Destinations

The **Destinations** section appears only when **Indexer discovery** is set to its `No` default. Here, you specify a known set of Splunk receivers on which to load-balance data.

Click **+ Add Destination** to specify more receivers on new rows. Each row provides the following fields:

- **Address**: Hostname of the Splunk receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

- **Port**: Port number to send data to.

- **TLS**: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

- **TLS servername**: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

- **Load weight**: The weight to apply to this Destination for load-balancing purposes.

The final column provides an `X` button to delete any row from the **Destinations** table.

Optional Settings

Toggle **Exclude current host IPs** to `Yes` if you want to exclude all the current host's IP addresses from the list of resolved hostnames.

Backpressure behavior: Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`. When toggled to `Persistent Queue`, adds the Persistent Queue Settings section (left tab) to the modal.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Enabling Cluster Manager Authentication

To enable token authentication on the Splunk cluster manager, you can find complete instructions in Splunk's Enable or Disable Token Authentication documentation. This option requires Splunk 7.3.0 or higher, and requires the following capabilites: `list_indexer_cluster` and `list_indexerdiscovery`.

For details on creating the token, see Splunk's Create Authentication Tokens topic – especially its section on how to Configure Token Expiry and "Not Before" Settings.

> Be sure to give the token an **Expiration** setting well in the future, whether you use **Relative Time** or **Absolute Time**. Otherwise, the token will inherit Splunk's default expiration time of `+30d` (30 days in the future), which will cause indexer discovery to fail.

If you have a failover site configured on Splunk's cluster manager, Cribl respects this configuration, and forwards the data to the failover site in case of site failure.

Persistent Queue Settings

> This 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`; `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: Toggle to `Yes` to reject certificates that are **not** authorized by a CA in the **CA certificate path**, nor by another trusted CA (e.g., the system's CA).

Server name (SNI): Server name for the SNI (Server Name Indication) TLS extension. This must be a host name, not an IP address.

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_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

Output Multi Metrics: Toggle this slider to `Yes` to output multiple-measurement metric data points. (Supported in Splunk 8.0 and above, this format enables sending multiple metrics in a single event, improving the efficiency of your Splunk capacity.)

Minimize in-flight data loss: If set to `Yes` (the default), Cribl Stream will check whether the indexer is shutting down and, if so, stop sending data. This helps minimize data loss during shutdown.

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.

Throttling: Throttle rate, in bytes per second. Multiple byte units such as KB, MB, GB, etc., are also allowed. E.g., `42 MB`. Default value of `0` indicates no throttling. When throttling is engaged, excess data will be dropped only if **Backpressure behavior** is set to **Drop events**. (Data will be blocked for all other **Backpressure behavior** settings.)

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

SSL Configuration for Splunk Cloud – Special Note

To connect to Splunk Cloud, you will need to extract the private and public keys from the Splunk-provided Splunk Cloud Universal Forwarder credentials package. You will also need to reference the CA Certificate located in the same package.

You can reuse many of the settings in this Splunk Cloud package to set up Splunk Cloud Destinations. Use the following steps:

Step 1. Extract the `splunkclouduf.spl` package on the Cribl 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:

```
openssl s_client -CA 100_<your-splunk-cloud>_splunkcloud/default/my-splunk-
cloud_cacert.pem -connect inputs1.<your-splunk-cloud>.splunkcloud.com:9997
```

To test the connection, you can use any of the URLs listed in the `[tcpout:splunkcloud]` stanza's `outputs.conf` section.

> You can simplify Steps 3 and 4 below by dragging and dropping (or uploading) the `.pem` files into Cribl 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>_server_cert.pem -out private.pem
```

If you are using RSA keys, instead use:

```
openssl rsa -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-
cloud>_server_cert.pem -out private.pem
```

Step 4. Extract the public key for the Server Certificate:

```
openssl x509 -in 100_<your-splunk-cloud>_splunkcloud/default/<your-splunk-
cloud>_server_cert.pem -out server.pem
```

Step 5. In the Splunk Load Balanced Destination's TLS Settings (Client Side) section, enter the following:

- CA Certificate Path: Path to `<your-splunk-cloud>_cacert.pem.`
- Private Key Path (mutual auth): Path to `private.pem` (Step 3 above).
- Certificate Path (mutual auth): Path to `server.pem` (Step 4 above).

> In a distributed deployment, enter this Destination configuration on each Worker Group/Fleet that forwards to Splunk Cloud. Then commit and deploy your changes.

Step 6. In a distributed deployment, enable Worker UI access, and verify that the Certificate files have been distributed to individual workers. If they are not present, copy the Certificate files to the Workers, using exactly the same paths you used at the Group level.

Notes About Forwarding to Splunk

- Data sent to Splunk is **not** compressed.

- The only `ack` from indexers that Cribl 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.

;

8.9. Internal

8.9.1. Cribl HTTP

The Cribl HTTP Destination is available only in distributed deployments. It enables Worker Nodes to send data to peer Nodes, as long as all Nodes are connected to the same Leader.

You'll find this Destination especially valuable in a hybrid Cloud deployment. In single-instance mode or for testing, you can substitute the Webhook Destination. (However, this substitution will not facilitate sending all internal fields, as described below.)

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

You might choose this Destination over the Cribl TCP Destination in certain circumstances, such as when a firewall or proxy blocks raw TCP egress.

Configuration Requirements

Configuring Cribl HTTP flow between peer Worker Nodes imposes some particular requirements:

- The Cribl HTTP Destination must be on a Worker Node that is connected to the same Leader as the Cribl HTTP Source(s).
- You must specify the same Leader Address on the Worker Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.
- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.
- To configure the Leader Address via the UI, log directly into each Worker Node's UI. Then select ⚙ **Settings** (lower left) > **Distributed Settings** > **Leader Settings** > **Address**.
- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Worker Nodes must be identical. In this example, both Worker Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- This Destination's **Cribl endpoint** field must point to the **Address** and **Port** you've configured on its peer Cribl HTTP Source(s).

Configuring a Cribl HTTP Destination

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Cribl HTTP**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 HTTP**. Next, click **+ Add New** to open a **Cribl HTTP** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Load balancing: Set to `No` by default. When toggled to `Yes`, see Load Balancing Settings below.

Cribl endpoint: URL of a Cribl Worker to send events to, e.g., `http://localhost:10200`.

> The **Cribl endpoint** field appears only when **Load balancing** is toggled to `Off`. Its value must point to the **Address** and **Port** you've configured on the peer Cribl HTTP Source to which you're sending.

Optional Settings

Compression: Codec to use to compress the data before sending. Defaults to `Gzip`.

Backpressure behavior: Specifies whether to block, drop, or queue events when all receivers are exerting backpressure. Defaults to `Block`. See Persistent Queue Settings below.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider displays the following controls.

Exclude Current Host IPs: This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

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**: Specify a weight to apply to this Worker, for load-balancing purposes.

- The final column provides an `X` button to delete any row from the table.

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_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

Validate server certs: Reject certificates that are not authorized by a CA in the **CA certificate path**, or by another trusted CA (e.g., the system's CA). Defaults to `Yes`.

Round-robin DNS: Toggle to `Yes` to use round-robin DNS lookup across multiple IPv6 addresses. When a DNS server returns multiple addresses, this will cause Cribl Stream to cycle through them in the order returned. Only displayed when the General Settings tab's **Load balancing** option is disabled.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Exclude fields: Fields to exclude from the event. By default, this Destination forwards all useful internal fields.

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`.

;

8.9.2. Cribl TCP

The Cribl TCP Destination is available only in distributed deployments. It enables Worker Nodes to send data to peer Nodes, as long as all Nodes are connected to the same Leader. You'll find this Destination especially valuable in a hybrid Cloud deployment.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

You might choose this Destination over the Cribl HTTP Destination in certain circumstances, such as when a firewall or proxy allows raw TCP egress. In single-instance mode or for testing, you can substitute the TCP JSON Destination. (However, this substitution will not facilitate sending all internal fields, as described below.)

Configuration Requirements

Configuring Cribl TCP flow between peer Worker Nodes imposes some particular requirements:

- The Cribl TCP Destination must be on a Worker Node that is connected to the same Leader as the Cribl TCP Source(s).

- You must specify the same Leader Address on the Worker Nodes that host both the Destination and Source. Otherwise, token verification will fail – breaking the connection and preventing data flow.

- To get the Leader Address specifically for Cribl.Cloud hybrid Workers, see Hybrid Cribl HTTP/Cribl TCP Configuration.

- To configure the Leader Address via the UI, log directly into each Worker Node's UI. Then select ⚙ **Settings** (lower left) > **Distributed Settings** > **Leader Settings** > **Address**.

- To configure the Leader Address via the instance.yml file, the `host` values on the connecting Worker Nodes must be identical. In this example, both Worker Nodes must point to `cribl-leader`:

```
distributed:
  mode: master
  master:
    host: cribl-leader
    port: 4200
```

- This Destination's **Address** and **Port** values must match the **Address** and **Port** you've configured on its peer Cribl TCP Source(s).

Configuring a Cribl TCP Destination

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Cribl TCP**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 TCP**. Next, click **+ Add New** to open a **Cribl TCP** > **New Destination** modal that provides the following options and fields.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** silder displays the following controls:

Exclude Current Host IPs

This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

Destinations

The **Destinations** table is where you specify a set of receivers on which to load-balance data. Click **+ Add Destination** to specify more receivers on new rows. Each row provides the following fields:

Address: Hostname of a receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on the host.

> Each **Address**/**Port** combination must match the **Address** and **Port** configured on a peer Cribl HTTP Source to which you're sending.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP); otherwise, uses the global TLS settings.

Load Weight: Specify a weight to apply to the receiver for load-balancing purposes.

The final column provides an `X` button to delete any row from the table.

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_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

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, this Destination forwards all useful internal fields.

Environment: If you're using GitOps, 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.

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`.

;

8.9.3. Cribl Stream (Deprecated)

> This Destination is deprecated as of Cribl Stream 3.5. 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 New** 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 **+ Add New** to open a **Cribl Stream** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Address: Hostname of the receiver.

Port: Port number to connect to on the host.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, you can optionally enter an auth token to use in the connection header.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Compression: Codec to use to compress the data before sending. Defaults to `None`.

Throttling: Throttle rate, in bytes per second. Defaults to `0`, meaning no throttling. Multiple-byte units such as KB, MB, GB etc. are also allowed, e.g., `42 MB`. When throttling is engaged, your **Backpressure behavior** selection determines whether Cribl 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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)

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.

;

8.9.4. Default

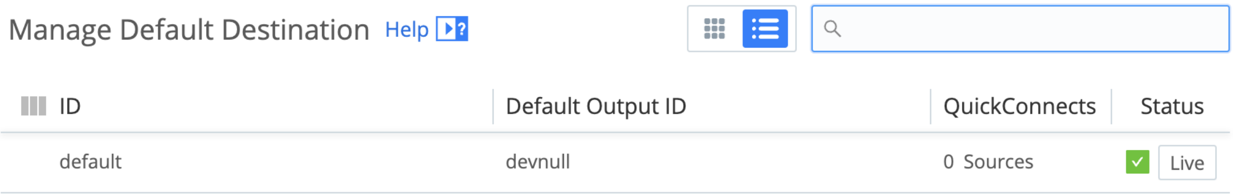
The **Default** Destination simply enables you to specify a default output from among your already-configured Destinations.

> Type: Internal | TLS Support: N/A | PQ Support: N/A

Configuring Cribl Stream's Default Destination

In the QuickConnect UI, 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.

In the **Data Routes** UI: From the top nav of a Cribl Stream instance or Group, select **Data** > **Destinations**, then select **Default** from the **Data Destinations** page's tiles or the **Destinations** left nav. 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.

;

8.9.5. 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 of a Cribl Stream instance or Group, select **Data** > **Destinations**, then select **DevNull** from the **Data Destinations** page's tiles or the **Destinations** left nav. Look for the **Live** indicator at the top right.

In the QuickConnect UI, a **DevNull** Destination is preconfigured and ready to use in the right **Destinations** column.

;

8.9.6. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Output Router**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Output Router**. Next, click **+ Add New** to open an **Output Router** > **New Destination** modal that provides the following options and fields.

General Settings

Router name: Enter a unique name to identify this Router definition.

Rules: A list of event routing rules. Each provides the following settings:

- **Filter expression**: JavaScript expression to select events to send to output.
- **Output**: Output to send matching events to.
- **Description**: Optionally, enter a description of this rule's purpose.
- **Final**: Toggle to `No` if you want the event to be checked against other rules lower in the stack.

Optional Settings

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl 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.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Advanced Settings

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Limitations/Options

- An Output Router cannot reference another. This is by design, so as 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 catchall rule to change this behavior.
- No post-processing (conditioning) can be done here. Instead, use pre-processing Pipelines on the Source tier.
- 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).
- Send the remaining events to `New York City`.

Router Name: **router66**

FILTER EXPRESSION	OUTPUT	FINAL
`host.startsWith('66')`	**San Francisco**	Yes

FILTER EXPRESSION	OUTPUT	FINAL
`method=='POST' || method=='GET`	**Seattle**	No
`method=='POST' || method=='GET'`	**Los Angeles**	Yes
`true`	**New York**	Yes

;

8.10. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Datadog**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Datadog**. Next, click **+ Add New** to open a **Datadog** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Destination definition.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays a field for you to enter an **API key** that is available in your Datadog profile.

- **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references the API access token described above. A **Create** link is available to store a new, reusable secret.

API key: Enter your Datadog organization's API key.

Optional Settings

Datadog site: Select the `US` (default) or `Europe` region.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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_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

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: Toggle this slider to `Yes` to compress log events' payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

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.

For More Information

You might find these Datadog references helpful:

- Submit Metrics
- Send Logs
- Metrics Types

;

8.11. DataSet

Cribl Stream can send log events to DataSet. This Destination sends batches of events, as JSON, to the DataSet API's addEvent method.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to DataSet

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **DataSet**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will provide the following options and fields.

Or, 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 **DataSet**. Next, click **+ Add New** to open a **New Destination** modal that provides the following options and fields.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB` .

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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_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

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: Defaults to `Yes`, to compress log events' payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Click **+ Add header** to define **Name**/**Value** pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl 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**.

;

8.12. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Elasticsearch**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Elasticsearch**. Next, click **+ Add New** to open an **Elasticsearch** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Elasticsearch Destination definition.

Load balancing: Set to `No` by default. When toggled to `Yes`, see Load Balancing Settings below.

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.

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: Set to `No` by default. 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.

Optional Settings

Type: Specify document type to use for events. An event's `__type` field can overwrite this value.

Backpressure behavior: Specify whether to block, drop, or queue events when all receivers are exerting backpressure. `Defaults to Block.`

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider displays the following controls:

Exclude Current Host IPs

This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

Bulk API URLs

The **Bulk API URLs** table is where you specify a known set of receivers on which to load-balance data. Click **+ Add URL** to specify more receivers on new rows. Each row provides the following fields:

URL: Specify the URL to an Elastic node to send events to – e.g., `http://elastic:9200/_bulk`

Load weight: Specify a weight to apply to the receiver for load-balancing purposes.

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.

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`; `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_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

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. (This option is visible only when the **General Settings** > **Load balancing** option is set to `No`.)

Compress: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Flush period (s): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Extra parameters: Name/Value pairs to pass as additional parameters. If you are using Elastic ingest pipelines, specify an extra parameter whose name is `pipeline` and whose value is the name of your pipeline, similar to these examples.

Elastic version: Determines how to format events. For Elastic Cloud, you must explicitly set version `7.x`. For other Elasticsearch clusters, the `Auto` default will discover the downstream Elasticsearch version automatically, but you have the option to explicitly set version `6.x` or `7.x`.

Elastic pipeline: To send data to an Elastic Ingest pipeline, optionally enter that pipeline's name as a constant. Or, enter a JavaScript expression that evaluates outgoing events, and sends matching events to the desired Elastic Ingest pipeline(s). For example, the expression `sourcetype=='access_common'?'cribl_pipeline':undefined` matches events whose `sourcetype` is `access_common`, and sends them to an Elastic Ingest pipeline named `cribl_pipeline`.

> The next two fields appear only when the **General Settings** > **Load balancing** option is set to `Yes`.

DNS resolution period (seconds): Re-resolve any hostnames each time this interval recurs, and pick up destinations from the A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds.

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`

How Does Load Balancing Work

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all URLs. 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 all URLs simultaneously, and the amount sent to each URL depends on these parameters:

1. Respective destination **weight**.
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.

> If multiple receivers are behind a hostname – i.e., multiple A records – then 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.
>
> 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.

Example

Suppose we have two receivers, 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., receiver A will start the interval with **60** and receiver 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.

Notes on HTTP-based Outputs

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname with load balancing disabled, Cribl 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.

;

8.13. 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

Configuring Cribl Stream to Output to Filesystem Destinations

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Filesystem**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Filesystem**. Next, click **+ Add New** to open a **Filesystem** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Filesystem definition.

Output location: Final destination for the output files.

Data format: The output data format defaults to `JSON`. `Raw` and `Parquet` are also available. Selecting `Parquet` (supported only on Linux, not Windows) exposes a **Parquet Settings** left tab, where you **must** configure certain options in order to export data in Parquet format.

Optional Settings

Staging location: Local filesystem location in which to buffer files before compressing and moving them to the final destination. Cribl recommends that this location be stable and high-performance. (This field is not displayed or available on Cribl.Cloud-managed 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 `none`. Cribl recommends setting this to `gzip`. This setting is not available when **Data format** is set to `Parquet`.

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to ``` `.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}` ```, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

Backpressure Behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

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.

Parquet Settings

To write out Parquet files, note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Parquet schema: Select a schema from the drop-down. The default `sample_parquet` schema is always available.

> Cribl recommends that you add a new schema – or clone the sample schema and modify it to suit your needs – via **Processing** > **Knowledge** > **Parquet Schemas**. Schemas that you add there will become available in this drop-down. For details, see Parquet Schemas.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

> Cribl 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: 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 slider'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.

;

8.14. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Honeycomb**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Honeycomb**. Next, click **+ Add New** to open a **Honeycomb** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Honeycomb definition.

Dataset name: Name of the dataset to send events to. (E.g., `iLoveObservabilityDataset`.)

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays a field for you to enter the **API key** for the team to which the dataset belongs.

- **Secret**: This option exposes an **API key (text secret)** drop-down, in which you can select a stored secret that references the API key described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, 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

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname, Cribl 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.

;

8.15. Humio HEC

The **Humio HEC** Destination can stream data to a Humio HEC (HTTP Event Collector) in JSON or Raw format.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Recommendations

To load-balance, Cribl recommends following Humio's Cluster Setup – place a load balancer in front of cluster nodes.

We recommend sending events with the `sourceType` field set to a Humio parser. This tells Humio which parser to use to extract fields (e.g., `"sourceType":"json"`).

If Humio cannot match the `sourceType` value to a parser, it will use the `kv` parser, and you will get an error that Humio 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 Humio HEC Destinations

In the **QuickConnect** UI: Click **+ Add Destination** on the right. From the resulting drawer's tiles, select **Humio HEC**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will provide the following options and fields.

Or, 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 **Humio HEC**. Next, click **+ Add New** to open a **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Humio HEC definition.

Humio HEC endpoint: URL of a Humio HEC 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 **Humio HEC endpoint** you specify above.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays a **HEC Auth token** field for you to enter your Humio HEC API token.

- **Secret**: Displays a **HEC Auth token (text secret)** drop-down, in which you can select a stored secret that references the API token described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This tab is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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_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

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 before blocking. This is set per Worker Process. Defaults to `5`. Each request can potentially hit a different HEC receiver.

Max body size (KB): Maximum size, in KB, of the request body. Defaults to `4096`. Lowering the size can potentially result in more parallel requests and also cause outbound requests to be made sooner.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1`.

> - Retries happen on this flush interval.
> - Any HTTP response code in the `2xx` range is considered success.
> - Any response code in the `5xx` range is considered a retryable error, which will not trigger Persistent Queue (PQ) usage.
> - Any other response code will trigger PQ (if PQ is configured as the Backpressure behavior).

Extra HTTP headers: Click **+ Add Header** to add **Name**/**Value** pairs to pass as additional HTTP headers.

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

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new connection will be reattempted. This is to prevent sticking to a particular Destination when there is a constant flow of events.

- If the server does not support keepalives – or if the server closes a pooled connection while idle – a new connection will be established for the next request.

- Cribl 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 Humio HEC servers.

;

8.16. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **InfluxDB**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **InfluxDB**. Next, click **+ Add New** to open an **InfluxDB** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this InfluxDB definition.

Write API URL: The URL of an InfluxDB cluster to send events to. (E.g., `http://localhost:8086/write`.)

Use v2 API: You can enable the InfluxDB v2 API with InfluxDB version 1.8 or later. This toggle defaults to `No` – which falls back to the v1 API, and displays a **Database** field.

If you toggle **Use v2 API** to `Yes`, Cribl 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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Use the **Authentication type** drop-down to select one of these options:

None: This default setting does not use authentication.

Auth token: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.

Auth token (text secret): This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.

Basic: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.

Basic (credentials secret): This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

;

8.17. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **MinIO**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **MinIO**. Next, click **+ Add New** to open a **MinIO** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this MinIO definition.

MinIO endpoint: MinIO service URL (e.g., http://minioHost:9000).

MinIO bucket name:Name of the destination MinIO bucket. This value can be a constant, or a JavaScript expression that will be evaluated only at init time. E.g., referencing a Global Variable: `myBucket-${C.vars.myVar}`. Ensure that the bucket already exists, otherwise MinIO will generate "bucket does not exist" errors.

> Event-level variables are not available for JavaScript expressions. This is because the bucket name is evaluated only at Destination initialization. If you want to use event-level variables in file paths, Cribl recommends specifying them in the **Partitioning Expression** field (described below), because this is evaluated for each file.

Staging location: Filesystem location in which to locally buffer files before compressing and moving to final destination. Cribl recommends that this location be stable and high-performance. (This field is not displayed or available on Cribl.Cloud-managed 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 `none`. Cribl recommends setting this to `gzip`. This setting is not available when **Data format** is set to `Parquet`.

File name prefix expression: The output filename prefix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to `CriblOut`.

File name suffix expression: The output filename suffix. Must be a JavaScript expression (which can evaluate to a constant), enclosed in quotes or backticks. Defaults to ``.${C.env["CRIBL_WORKER_ID"]}.${__format}${__compression === "gzip" ? ".gz" : ""}``, where `__format` can be `json` or `raw`, and `__compression` can be `none` or `gzip`.

Backpressure behavior: Select whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

How MinIO Composes File Names

The full path to a file consists of:

```
<bucket_name>/<keyprefix><partition_expression | __partition><file_name_prefix><filename>.<extension>
```

As an example, assume that the **MinIO bucket name** is `bucket1`, the **Key prefix** is `aws`, the **Partitioning expression** is `` `${host}/${sourcetype}` ``, the source is undefined, the **File name prefix** is the default `CriblOut`, and the **Data format** is `json`. Here, the full path as displayed in MinIO would have this form: `/bucket1/aws/192.168.1.241/undefined/CriblOut-<randomstring>0.json`

> Although MinIO will display the **Key prefix** and **Partitioning expression** values as folders, both are actually just part of the overall key name, along with the file name.

Authentication

Use the **Authentication Method** buttons to select one of these options:

Auto: This default option uses the AWS instance's metadata service to automatically obtain short-lived credentials from the IAM role attached to an EC2 instance. 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. 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_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.

Parquet Settings

To write out Parquet files, note that:

- Cribl Edge Workers support Parquet only when running on Linux, not on Windows.
- See Working with Parquet for pointers on how to avoid problems such as data mismatches.

Parquet schema: Select a schema from the drop-down. The default `sample_parquet` schema is always available.

> Cribl recommends that you add a new schema – or clone the sample schema and modify it to suit your needs – via **Processing** > **Knowledge** > **Parquet Schemas**. Schemas that you add there will become available in this drop-down. For details, see Parquet Schemas.

Row group size: Set the target memory size for row group segments. Modify this value to optimize memory use when writing. Value must be a positive integer smaller than the **File size** value, with appropriate units. Defaults to `16 MB`.

Page size: Set the target memory size for page segments. Generally, set lower values to improve reading speed, or set higher values to improve compression. Value must be a positive integer smaller than the **Row group size** value, with appropriate units. Defaults to `1 MB`.

Log invalid rows: Toggle to `Yes` to output up to 20 unique rows that were skipped due to data format mismatch. Log level must be set to `debug` for output to be visible.

Advanced Settings

Max file size (MB): Maximum uncompressed output file size. Files of this size will be closed and moved to final output location. Defaults to `32`.

Max file open time (sec): Maximum amount of time to write to a file. Files open for longer than this limit will be closed and moved to final output location. Defaults to `300`.

Max file idle time (sec): Maximum amount of time to keep inactive files open. Files open for longer than this limit will be closed and moved to final output location. Defaults to `30`.

Max open files: Maximum number of files to keep open concurrently. When exceeded, the oldest open files will be closed and moved to final output location. Defaults to `100`.

> Cribl Stream will close files when **either** of the `Max file size (MB)` or the `Max file open time (sec)` conditions is met.

Add Output ID: When set to `Yes` (the default), adds the **Output ID** field's value to the staging location's file path. This ensures that each Destination's logs will write to its own bucket.

> 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 slider'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.

Internal Fields

Cribl Stream uses a set of internal fields to assist in forwarding data to a Destination.

Field for this Destination:

- `__partition`

;

8.18. OpenTelemetry (OTel)

Cribl Stream supports sending events to 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.

When configuring Pipelines (including pre-processing and post-processing Pipelines), you need to ensure that events sent to this Destination conform to the relevant Protobuf specification:

- For traces, opentelemetry-proto/trace.proto at v0.9.0 · open-telemetry/opentelemetry-proto
- For metrics, opentelemetry-proto/metrics.proto at v0.9.0 · open-telemetry/opentelemetry-proto

The OTel Destination will drop non-conforming events. Also, when trying to convert an event to OTLP, if the Destination encounters a parsing error, it discards the event, and Cribl Streams logs the error.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output to OTel

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **OpenTelemetry**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **OpenTelemetry**. Next, click **+ Add New** to open an **OpenTelemetry** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this OTel output definition.

Endpoint: Where to send events, in any of a variety of formats (FQDN, PQDN, IP address and port, etc). The same endpoint is used for both Traces and Metrics. If no port is specified, we default to the standard port for OTel Collectors, 4137, unless TLS is enabled or the protocol is HTTPS.

Optional Settings

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

TLS Settings (Client Side)

Enabled 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).

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 section is displayed when the **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped and data blocking is applied. Enter a numeral with units of KB, MB, etc.

Queue file path: The location for the persistent queue files. This will be of the form: `your/path/here/<worker-id>/<output-id>`. Defaults to: `$CRIBL_HOME/state/queues`.

Compression: Codec to use to compress the persisted data, once a file is closed. Defaults to `None`. `Gzip` is also available.

Queue-full behavior: Whether to block or drop events when the queue is exerting backpressure (because disk is low or at full capacity). **Block** is the same behavior as non-PQ blocking, corresponding to the **Block** option on the **Backpressure behavior** drop-down. **Drop new data** throws away incoming data, while leaving the contents of the PQ unchanged.

Clear persistent queue: Click this button if you want to flush out files that are currently queued for delivery to this Destination. A confirmation modal will appear. (Appears only after **Output ID** has been defined.)

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_host` (Cribl Stream Node that processed the event).

Other options include:

- `cribl_pipe` – Cribl Stream Pipeline that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_wp` — Cribl Stream Worker Process that processed the event.

Advanced Settings

Connection Timeout: Amount of time (milliseconds) to wait for the connection to establish before retrying. Defaults to `10000`.

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`.

Keep Alive Time (seconds): How often the sender should ping the peer to keep the connection alive. Defaults to `30`.

Max body size (KB): Maximum size of the request body. Defaults to `4096`.

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`.

Metadata: Extra information to send with each gRPC request in the form of a list of key-value pairs. Value supports JavaScript expressions that are evaluated just once, when the destination gets started. If passing credentials as metadata, using `C.Secret` is recommended.

Environment: Optionally, specify a single Git branch on which to enable this configuration. If this field is empty, the config will be enabled everywhere.

;

8.19. SignalFx

Cribl Stream supports sending events to SignalFx.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to SignalFx

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **SignalFx**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **SignalFx**. Next, click **+ Add New** to open a **SignalFx** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this SignalFx definition.

Realm: SignalFx realm name (e.g., `us0`). Required.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays an **Auth token** field for you to enter your SignalFx API access token. See SignalFx's Manage Tokens topic.
- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the API access token described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **+ Add Header** to insert extra headers as **Name**/**Value** pairs.

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 About SignalFx

For details on integrating with SignalFx, see the SignalFx Developers Guide, with particular reference to the SignalFx HTTP Send Metrics Reference.

;

8.20. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **SNMP Trap**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **SNMP Trap**. Next, click **+ Add New** to open an **SNMP Trap** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this SNMP Trap definition.

SNMP Trap destinations: Click **+ Add Destination** to specify more SNMP destinations to forward traps to, on new rows. Each row provides the following fields:

- **Address**: Destination host.
- **Port**: Destination port. Defaults to `162` .

Optional Settings

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Processing Settings

Post-Processing

Pipeline: Pipeline to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl 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.

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.

;

8.21. Sumo Logic

Cribl Stream can send log and metric events to Sumo Logic over HTTP.

> Type: Streaming | TLS Support: Configurable | PQ Support: Yes

Configuring Cribl Stream to Output to Sumo Logic

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Sumo Logic**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Sumo Logic**. Next, click **+ Add New** to open a **Sumo Logic** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Sumo Logic Destination definition.

API URL: Enter the URL of the Sumo Logic HTTP Source to which events should be sent. (E.g., `https://endpoint6.collection.us2.sumologic.com/receiver/v1/http/<long-hash>`.)

Optional Settings

Custom source name: Optionally, override the source `name` configured on the Sumo Logic HTTP Source. This value will be sent with events via the `X-Sumo-Name` HTTP header.

Custom source category: Optionally, override the source `category` configured on the Sumo Logic Collector and/or HTTP Source. This value will be sent with events via the `X-Sumo-Category` HTTP header.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle this slider to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass as additional HTTP headers.

Failed request logging mode: Use this drop-down to determine which data should be logged when a request fails. Select among `None` (the default), `Payload`, or `Payload + Headers`. With this last option, Cribl Stream will redact all headers, except non-sensitive headers that you declare below in **Safe headers**.

Safe headers: Add headers to declare them as safe to log in plaintext. (Sensitive headers such as `authorization` will always be redacted, even if listed here.) Use a tab or hard return to separate header names.

Environment: If you're using GitOps, 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 Sumo Logic as a metric event. Otherwise, Cribl Stream will send it as a log event.

Notes on HTTP-based Outputs

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.
- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.
- When resolving the Destination's hostname, Cribl 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.

;

8.22. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Syslog**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Syslog**. Next, click **+ Add New** to open a **Syslog** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Syslog definition.

Protocol: The network protocol to use for sending out syslog messages. Defaults to `TCP`; `UDP` is also available.

Load balancing: This option is displayed only when the **Protocol** is set to `TCP`. When toggled to `Yes`, see Load Balancing Settings below.

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. Defaults to `Syslog`.

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

> Load balancing is available only when the **Protocol** is set to `TCP`.

Enabling the **Load balancing** slider replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude Current Host IPs

This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

Destinations

The **Destinations** table is where you specify a known set of receivers on which to load-balance data. Click **+ Add Destination** to specify more receivers on new rows. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: The weight to apply to this receiver for load-balancing purposes.

The final column provides an `X` button to delete any row from the table.

Persistent Queue Settings

> This section is displayed only for **TCP**, and only when **Backpressure behavior** is set to **Persistent Queue**.

Max file size: The maximum data volume to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to `1 MB`.

Max queue size: The maximum amount of disk space the queue is allowed to consume. Once this limit is reached, Cribl 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)

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 `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

> 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_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

The first two options are always displayed:

Octet count framing: When enabled, Cribl Stream prefixes messages with their byte count. When disabled, Cribl Stream omits prefixes, and instead appends a `\n` to messages.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

The following options are added if you enable the General Settings tab's **Load balancing** option:

DNS resolution period (seconds): Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to `600` seconds.

Load balance stats period (seconds): Lookback traffic history period. Defaults to `300` seconds. (Note that If multiple receivers are behind a hostname – i.e., multiple A records – all resolved IPs will inherit the weight of the host, unless each IP is specified separately. In Cribl 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.

How Does Load Balancing Work

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all receivers (listed as Destinations in the GUI). If FQDNs/hostnames are used as the Destination address and each resolves to, for example, 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 all receivers simultaneously, and the amount sent to each receiver depends on these parameters:

1. Respective destination **weight**.
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.

Example

Suppose we have two receivers, 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., receiver A will start the interval with **60** and receiver 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.

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 5424 and/or RFC 5425) before it is transmitted to downstream services.

The Syslog Destination's **General Settings** page offers several settings to format the timestamps, to format the message delivering the event, and to set the syslog-specific default settings for Facility, Severity, and App Name.

Below are two examples of RFC-compliant syslog events:

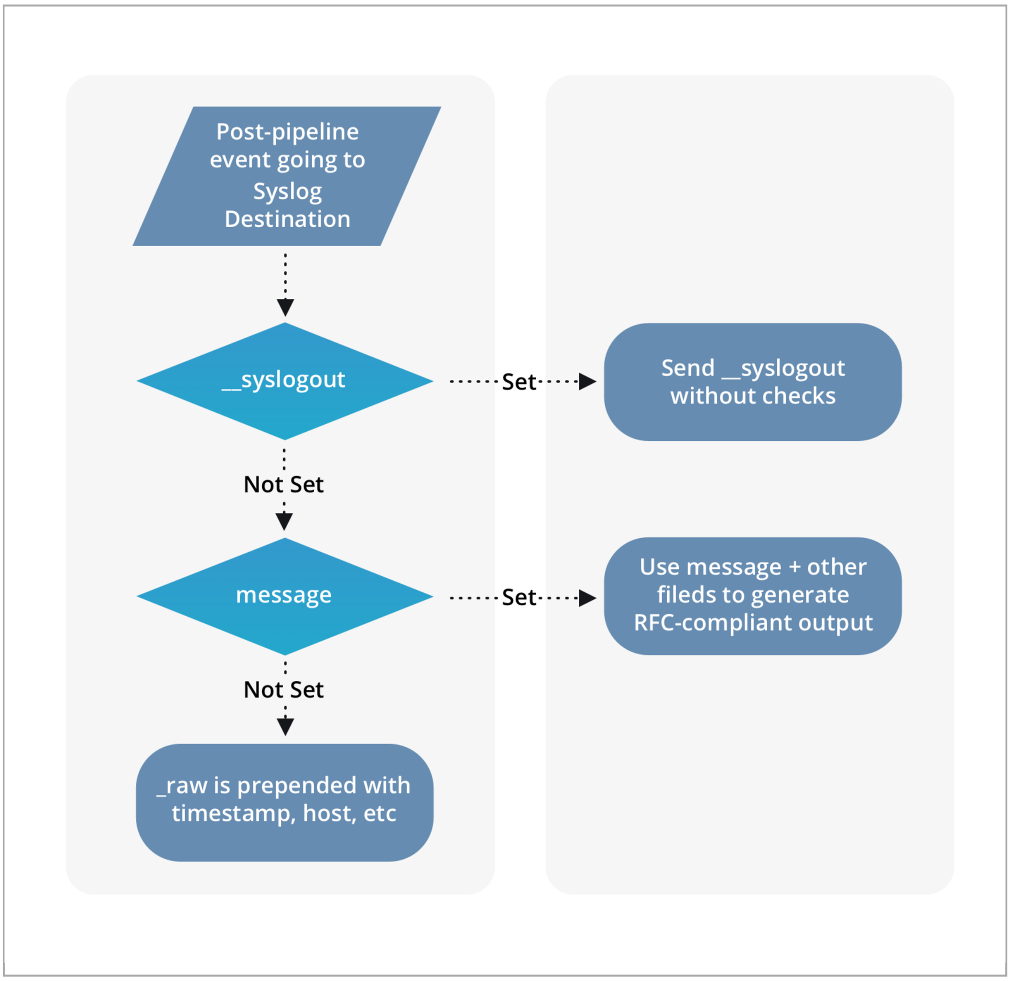
- `<13>Jul 11 10:34:35 testbox testing[42]`
- `<214>1 2022-07-11T18:58:45.000Z testbox testing[42]: foo=bar this=that base=ball gizmo=sprocket`

Cribl 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 fileds 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.

The most common uses for the `__syslogout` method are:

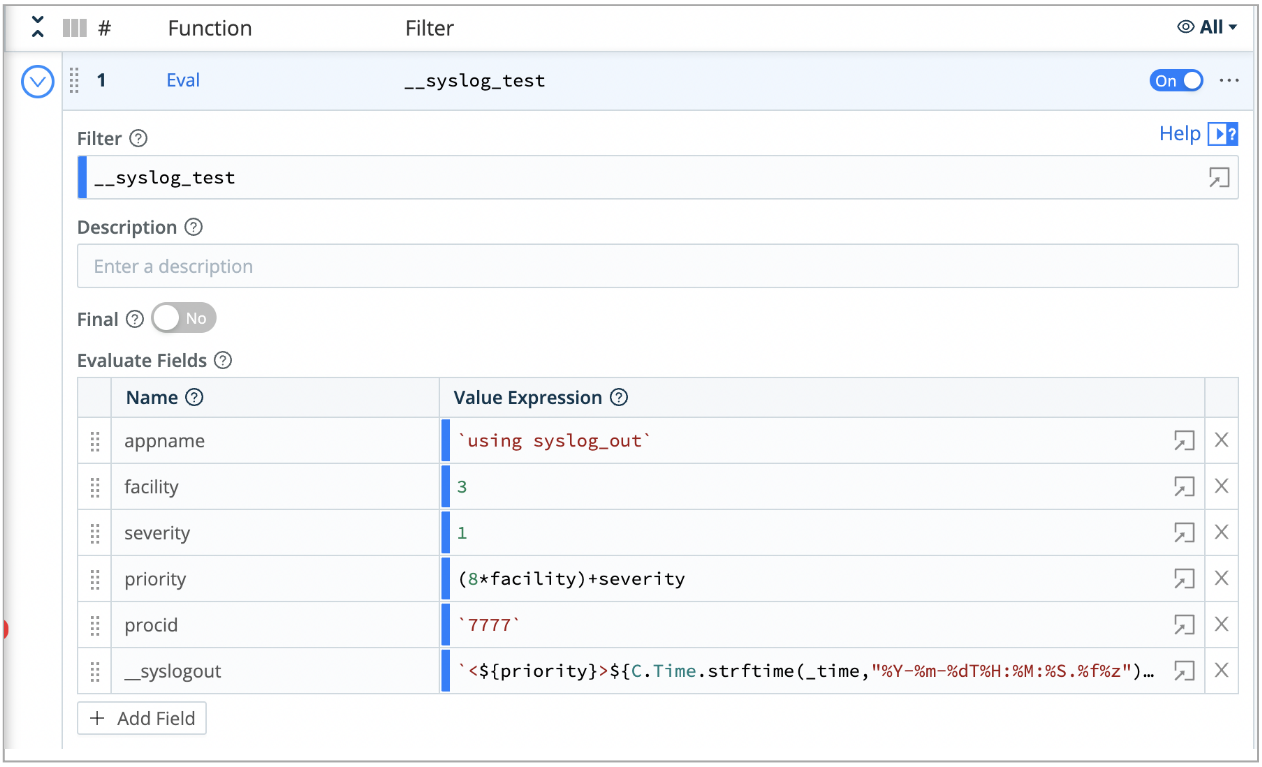
- When the value of `_raw` is already in syslog format as it comes in, and minimal processing is necessary.
- When sending raw TCP data to a destination that doesn’t enforce syslog RFCs, such as a raw TCP listener. Cribl Stream currently does not provide a native "raw TCP" Destination. However, in some 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:



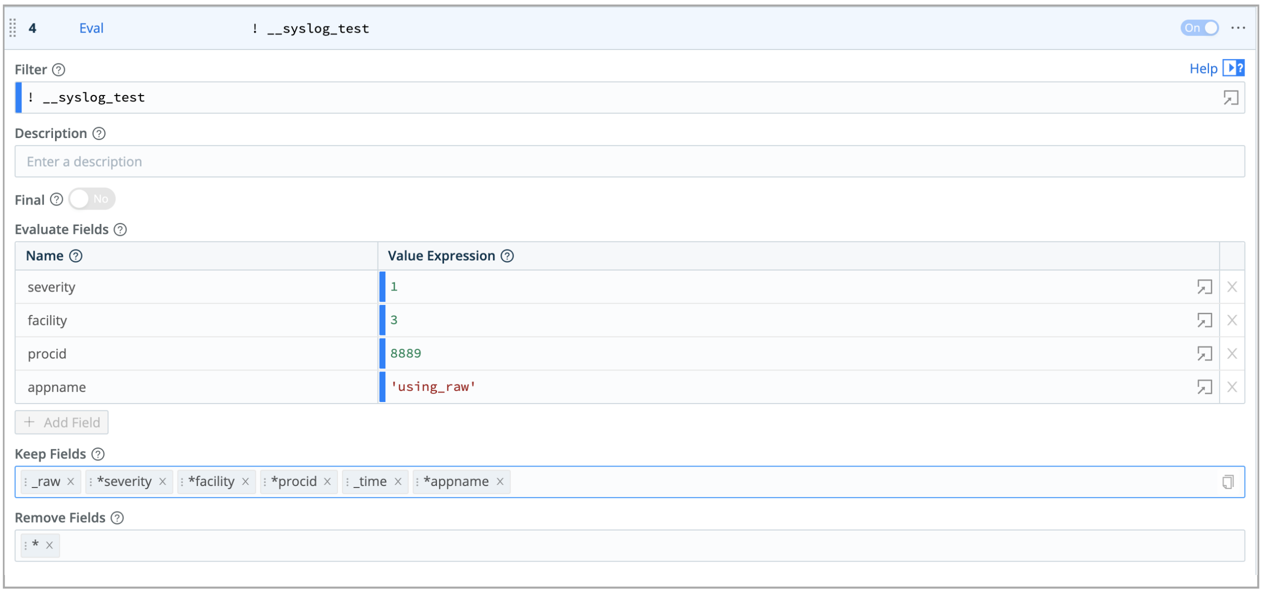
Name	Value Expression
appname	`` `using syslog_out` ``
facility	3
severity	1
priority	(8*facility)+severity
procid	`` `7777` ``
__syslogout	`` `<${priority}>${C.Time.strftime(_time,"%Y-%m-%dT%H:%M:%S.%f%z")… ``

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:



4 Eval ! __syslog_test

Filter: `! __syslog_test`

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
severity	1
facility	3
procid	8889
appname	'using_raw'

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\")}
${host} ${appname}[${procid}]: ${_raw}`"
            }
          ],
          "keep": [],
          "remove": []
        },
        "id": "eval",
        "disabled": false
      },
      {
        "filter": "! __syslog_test",
        "conf": {
          "add": [
            {
              "name": "severity",
              "value": "1"
            },
            {
              "name": "facility",
              "value": "3"
            },
            {
              "name": "procid",
              "value": "8889"
            },
            {
              "name": "appname",
              "value": "'using_raw'"
            }
          ],
          "keep": [
            "_raw",
            "*severity",
            "*facility",
            "*procid",
            "_time",
            "*appname"
          ],
          "remove": [
            "*"
          ]
        },
        "id": "eval",
        "disabled": false
      }
    ]
  }
}
```

In this method, Cribl 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.

;

8.23. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **TCP JSON**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **TCP JSON**. Next, click **+ Add New** to open a **TCP JSON** > **New Destination** modal that provides the following options and fields.

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 the default `No` setting.

Address: Hostname of the receiver.

Port: Port number to connect to on the host.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: In the resulting **Auth token** field, you can optionally enter an auth token to use in the connection header.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the `authToken` header field value described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

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 for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Load Balancing Settings

Enabling the **Load balancing** slider replaces the static **General Settings** > **Address** and **Port** fields with the following controls:

Exclude Current Host IPs

This slider determines whether to exclude all IPs of the current host from the list of any resolved hostnames. Defaults to `No`.

Destinations

The **Destinations** table is where you specify a known set of receivers on which to load-balance data. Click **+ Add Destination** to specify more receivers on new rows. Each row provides the following fields:

Address: Hostname of the receiver. Optionally, you can paste in a comma-separated list, in `<host>:<port>` format.

Port: Port number to send data to on this host.

TLS: Whether to inherit TLS configs from group setting, or disable TLS. Defaults to `inherit`.

TLS servername: Servername to use if establishing a TLS connection. If not specified, defaults to connection host (if not an IP). Otherwise, uses the global TLS settings.

Load weight: The weight to apply to this receiver for load-balancing purposes.

The final column provides an `X` button to delete any row from the table.

> 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.

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`; `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_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.

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.

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.

How Does Load Balancing Work

Cribl Stream will attempt to load-balance outbound data as fairly as possibly across all receivers (listed as Destinations in the GUI). If FQDNs/hostnames are used as the Destination address and each resolves to, for example, 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 all receivers simultaneously, and the amount sent to each receiver depends on these parameters:

1. Respective destination **weight**.
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.

Example

Suppose we have two receivers, 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., receiver A will start the interval with **60** and receiver 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.

;

8.24. Wavefront

Cribl Stream supports sending events to Wavefront analytics.

> Type: Streaming | TLS Support: Yes | PQ Support: Yes

Configuring Cribl Stream to Output to Wavefront

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Wavefront**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Wavefront**. Next, click **+ Add New** to open a **Wavefront** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this Wavefront definition.

Domain name: WaveFront domain name, e.g., `longboard`. Required.

Authentication Settings

Use the **Authentication method** buttons to select one of these options:

- **Manual**: Displays an **API key** field for you to enter your Wavefront API auth token. See Wavefront's Generating an API Token topic.

- **Secret**: This option exposes an **Auth token (text secret)** drop-down, in which you can select a stored secret that references the API auth token described above. A **Create** link is available to store a new, reusable secret.

Optional Settings

Backpressure behavior: Select whether to block, drop, or queue events when all receivers are exerting backpressure. (Causes might include a broken or denied connection, or a rate limiter.) Defaults to `Block`.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle to `Yes` to compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB.

Flush period (sec): Maximum time between requests. Low values can cause the payload size to be smaller than the configured **Max body size**. Defaults to `1` second.

Extra HTTP headers: Click **+ Add Header** to insert extra headers as **Name**/**Value** pairs.

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 About Wavefront

For details on integrating with Wavefront, see these Wavefront resources:

- Direct Data Ingestion, and adjacent topics on Wavefront Proxies.
- Wavefront Data Format.

;

8.25. 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

In the **QuickConnect** UI: Click **+ Add Destination** at right. From the resulting drawer's tiles, select **Webhook**. Next, click either **+ Add New** or (if displayed) **Select Existing**. The resulting drawer will 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** > **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 **Webhook**. Next, click **+ Add New** to open a **Webhook** > **New Destination** modal that provides the following options and fields.

General Settings

Output ID: Enter a unique name to identify this HTTP output definition.

URL: Endpoint URL to send events to. The internal field `__url`, where present in events, will override this value. See Internal Fields below.

Method: The HTTP verb to use when sending events. Defaults to `POST`. Change this to `PUT` or `PATCH` where required by the endpoint.

Format: The format in which to send out events. One of:

- `NDJSON (newline-delimited JSON)`: The default.
- `JSON Array`: Arrays in JSON-parseable format.
- `Custom`: Exposes the following additional fields to define the output format:
    - **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` – i.e., the value of the `__httpOut` field. Use the button at right to open a validation modal.

  - **Drop when null**: If toggled to `Yes`, Cribl 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.

Backpressure behavior: Whether to block, drop, or queue events when all receivers are exerting backpressure.

Tags: Optionally, add tags that you can use for filtering and grouping at the final destination. Use a tab or hard return between (arbitrary) tag names.

Authentication

Select one of the following options for authentication:

- **None**: Don't use authentication.

- **Auth token**: Use HTTP token authentication. In the resulting **Token** field, enter the bearer token that must be included in the HTTP authorization header.

- **Auth token (text secret)**: This option exposes a **Token (text secret)** drop-down, in which you can select a stored text secret that references the bearer token described above. A **Create** link is available to store a new, reusable secret.

- **Basic**: Displays **Username** and **Password** fields for you to enter HTTP Basic authentication credentials.

- **Basic (credentials secret)**: This option exposes a **Credentials secret** drop-down, in which you can select a stored text secret that references the Basic authentication credentials described above. A **Create** link is available to store a new, reusable secret.

Persistent Queue Settings

> This 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`; `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_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

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: Toggle this slider to `Yes` to gzip-compress the payload body before sending.

Request timeout: Amount of time (in seconds) to wait for a request to complete before aborting it. Defaults to `30`.

Request concurrency: Maximum number of concurrent requests before blocking. This is set per Worker Process. Defaults to `5`.

Max body size (KB): Maximum size of the request body. Defaults to `4096` KB. You can set this limit to as high as 500 MB (`512000` KB). Be aware that high values can cause high memory usage per Worker Node, especially if a dynamically constructed URL (see Internal Fields) causes this Destination to send events to more than one URL.

Max events per request: Maximum number of events to include in the request body. The `0` default allows unlimited events.

Flush period (sec): Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.

Extra HTTP headers: Name/Value pairs to pass 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.

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 send it as a log event.

If an event contains the internal field `__url`, that field's value will override the **General Settings** > **URL** value. This way, you can determine the endpoint URL dynamically.

For example, you could create a Pipeline containing an Eval Function that adds the `__url` field, and connect that Pipeline to your Webhook Destination. Configure the Eval Function to set `__url`'s value to a URL that varies depending on a global variable, or on some property of the event, or on some other dynamically generated value that meets your needs.

Use Cases

See these examples of configuring a Webhook Destination to integrate with specific services:

- Webhook/BigPanda Integration
- Webhook/Sumo Logic Integration

Notes on HTTP-based Outputs

- Cribl Stream will attempt to use keepalives to reuse a connection for multiple requests. After 2 minutes of the first use, the connection will be thrown away, and a new one will be reattempted. This is to prevent sticking to a particular destination when there is a constant flow of events.

- If the server does not support keepalives (or if the server closes a pooled connection while idle), a new connection will be established for the next request.

- When resolving the Destination's hostname, Cribl 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.

;

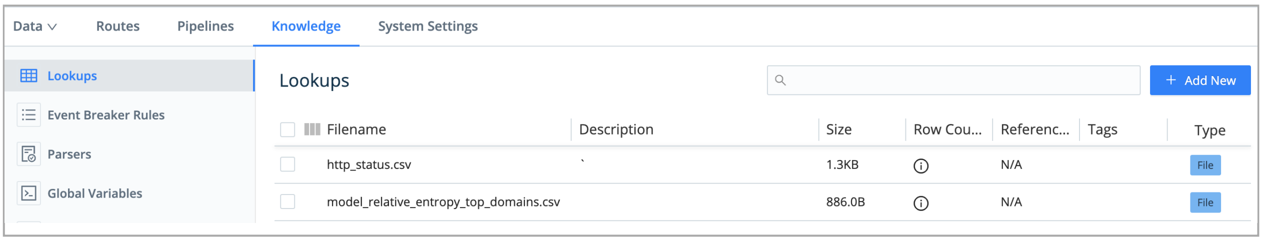
9. Knowledge

9.1. 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.



Filename	Description	Size	Row Cou...	Referenc...	Tags	Type
http_status.csv	`	1.3KB	ⓘ	N/A		File
model_relative_entropy_top_domains.csv		886.0B	ⓘ	N/A		File

Lookups Library

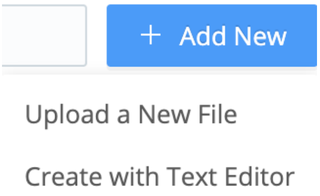
How Does the Library Work

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 **+ Add New**, then click the appropriate option from the drop-down.

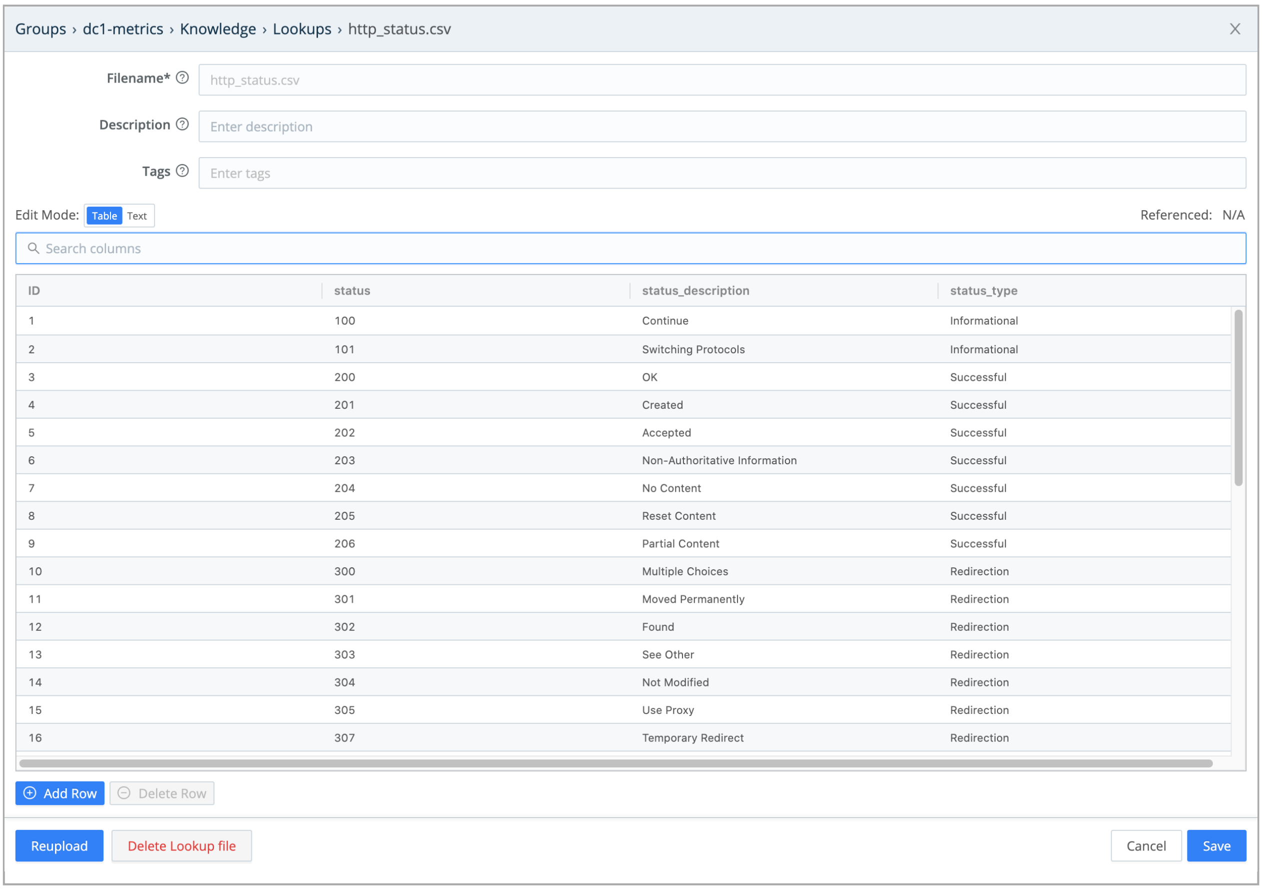


Adding a lookup file

Modifying Lookup Files

To re-upload, expand, edit, or delete an existing `.csv` or `.gz` lookup file, click its row on the Lookups page. (No editing option is available for `.mmdb` files; you can only re-upload or delete these.)

In the resulting modal, you can edit files in **Table** or **Text** mode. However, **Text** mode is disabled for files larger than 1 MB.


Groups › dc1-metrics › Knowledge › Lookups › http_status.csv
×

Filename* ⓘ http_status.csv

Description ⓘ Enter description

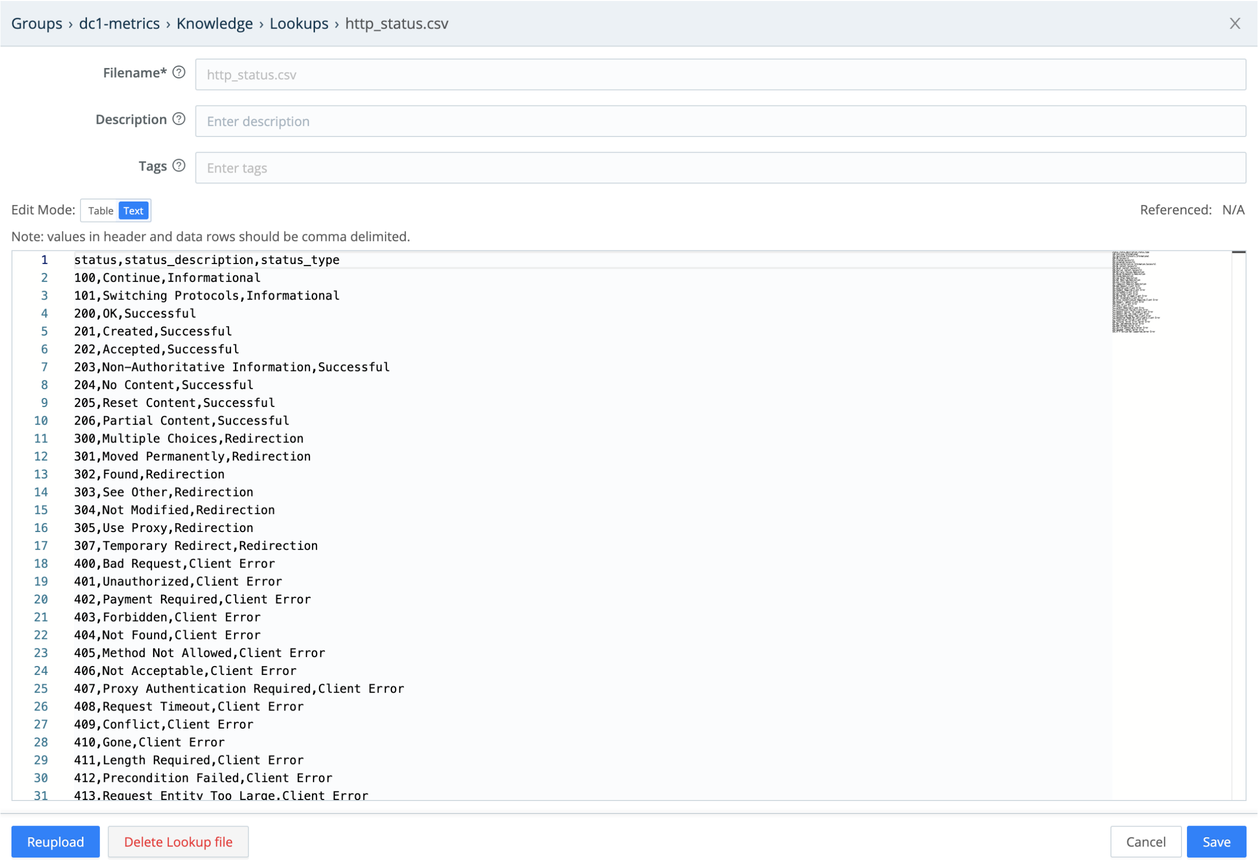
Tags ⓘ Enter tags

Edit Mode: Table Text

Referenced: N/A

ID	status	status_description	status_type
1	100	Continue	Informational
2	101	Switching Protocols	Informational
3	200	OK	Successful
4	201	Created	Successful
5	202	Accepted	Successful
6	203	Non-Authoritative Information	Successful
7	204	No Content	Successful
8	205	Reset Content	Successful
9	206	Partial Content	Successful
10	300	Multiple Choices	Redirection
11	301	Moved Permanently	Redirection
12	302	Found	Redirection
13	303	See Other	Redirection
14	304	Not Modified	Redirection
15	305	Use Proxy	Redirection
16	307	Temporary Redirect	Redirection

Editing in table mode


Groups › dc1-metrics › Knowledge › Lookups › http_status.csv
Edit Mode: Table Text
Reupload
Delete Lookup file
Cancel
Save

Filename*: http_status.csv

Description: Enter description

Tags: Enter tags

Referenced: N/A

Note: values in header and data rows should be comma delimited.

```
status,status_description,status_type
100,Continue,Informational
101,Switching Protocols,Informational
200,OK,Successful
201,Created,Successful
202,Accepted,Successful
203,Non-Authoritative Information,Successful
204,No Content,Successful
205,Reset Content,Successful
206,Partial Content,Successful
300,Multiple Choices,Redirection
301,Moved Permanently,Redirection
302,Found,Redirection
303,See Other,Redirection
304,Not Modified,Redirection
305,Use Proxy,Redirection
307,Temporary Redirect,Redirection
400,Bad Request,Client Error
401,Unauthorized,Client Error
402,Payment Required,Client Error
403,Forbidden,Client Error
404,Not Found,Client Error
405,Method Not Allowed,Client Error
406,Not Acceptable,Client Error
407,Proxy Authentication Required,Client Error
408,Request Timeout,Client Error
409,Conflict,Client Error
410,Gone,Client Error
411,Length Required,Client Error
412,Precondition Failed,Client Error
413,Request Entity Too Large,Client Error
```

Editing in text mode

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).

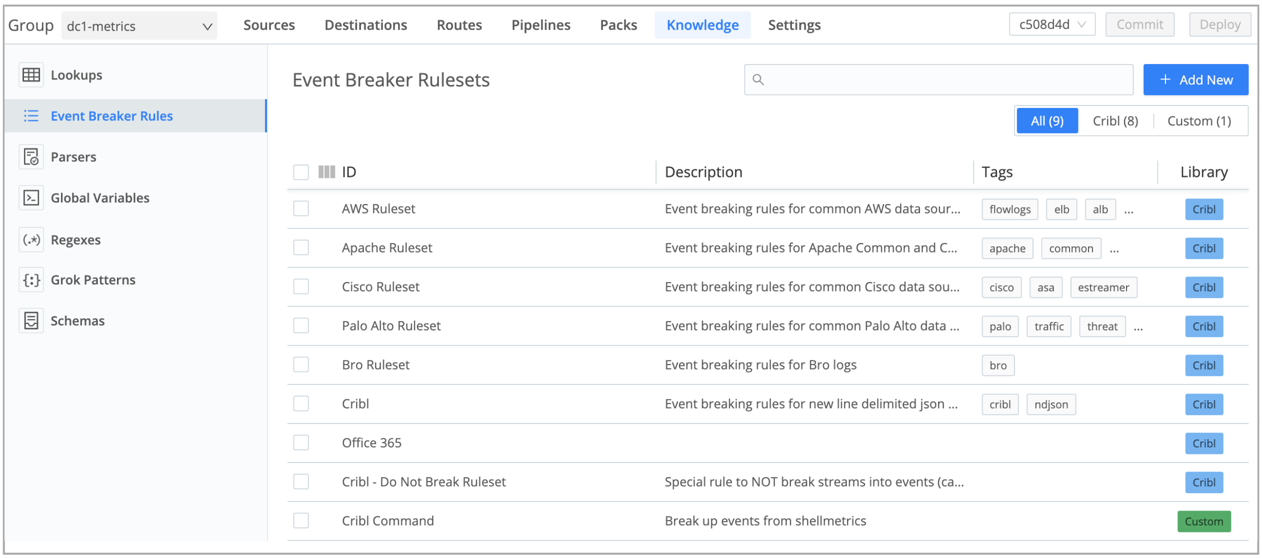
;

9.2. 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 Breakers**. On the resulting **Event Breaker Rulesets** page, you can edit, add, delete, search, and tag Event Breaker rules and rulesets, as necessary.



Event Breaker Rulesets

ID	Description	Tags	Library
AWS Ruleset	Event breaking rules for common AWS data sour...	flowlogs elb alb ...	Cribl
Apache Ruleset	Event breaking rules for Apache Common and C...	apache common ...	Cribl
Cisco Ruleset	Event breaking rules for common Cisco data sou...	cisco asa estreamer	Cribl
Palo Alto Ruleset	Event breaking rules for common Palo Alto data ...	palo traffic threat ...	Cribl
Bro Ruleset	Event breaking rules for Bro logs	bro	Cribl
Cribl	Event breaking rules for new line delimited json ...	cribl ndjson	Cribl
Office 365			Cribl
Cribl - Do Not Break Ruleset	Special rule to NOT break streams into events (ca...		Cribl
Cribl Command	Break up events from shellmetrics		Custom

Event Breaker Rulesets page

You can use Cribl Edge to apply and author Event Breakers while exploring files – even files that you have not made into sample files. This includes remote files that are viewable only from Edge. See Exploring Cribl Edge.

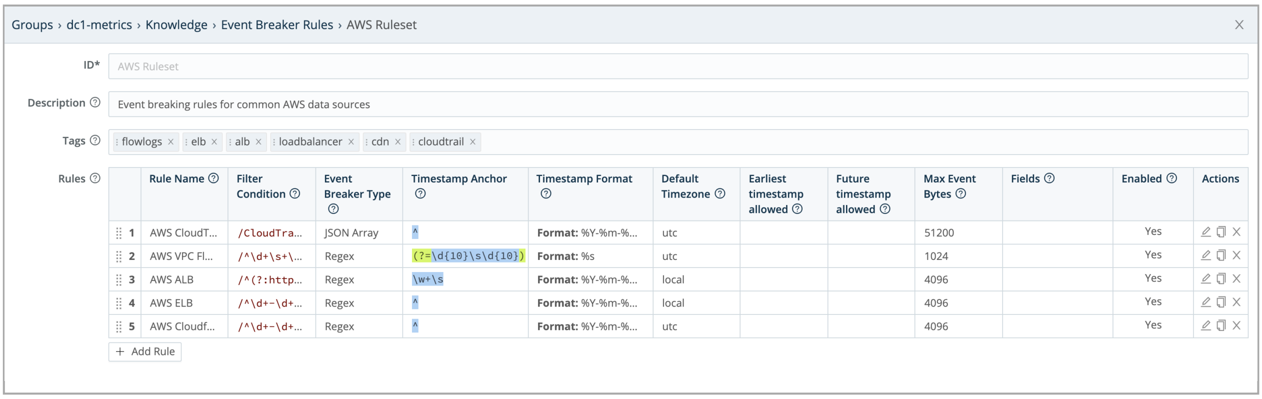
Limitations

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.



Groups › dc1-metrics › 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

Rules:

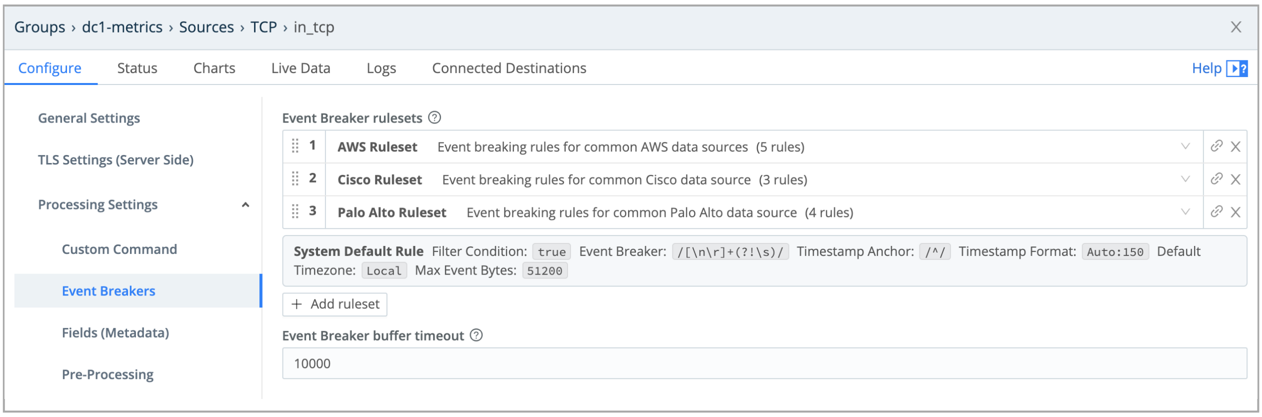
	Rule Name	Filter Condition	Event Breaker Type	Timestamp Anchor	Timestamp Format	Default Timezone	Earliest timestamp allowed	Future timestamp allowed	Max Event Bytes	Fields	Enabled	Actions
1	AWS CloudT...	/CloudTra...	JSON Array	^	**Format:** %Y-%m-%...	utc			51200		Yes	
2	AWS VPC Fl...	/^\d+\s+\...	Regex	(?=\d{10}\s\d{10})	**Format:** %s	utc			1024		Yes	
3	AWS ALB	/^(?:http...	Regex	\w+\s	**Format:** %Y-%m-%...	local			4096		Yes	
4	AWS ELB	/^\d+-\d+...	Regex	^	**Format:** %Y-%m-%...	local			4096		Yes	
5	AWS Cloudf...	/^\d+-\d+...	Regex	^	**Format:** %Y-%m-%...	utc			4096		Yes	

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.

```
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:



Groups › dc1-metrics › Sources › TCP › in_tcp

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)

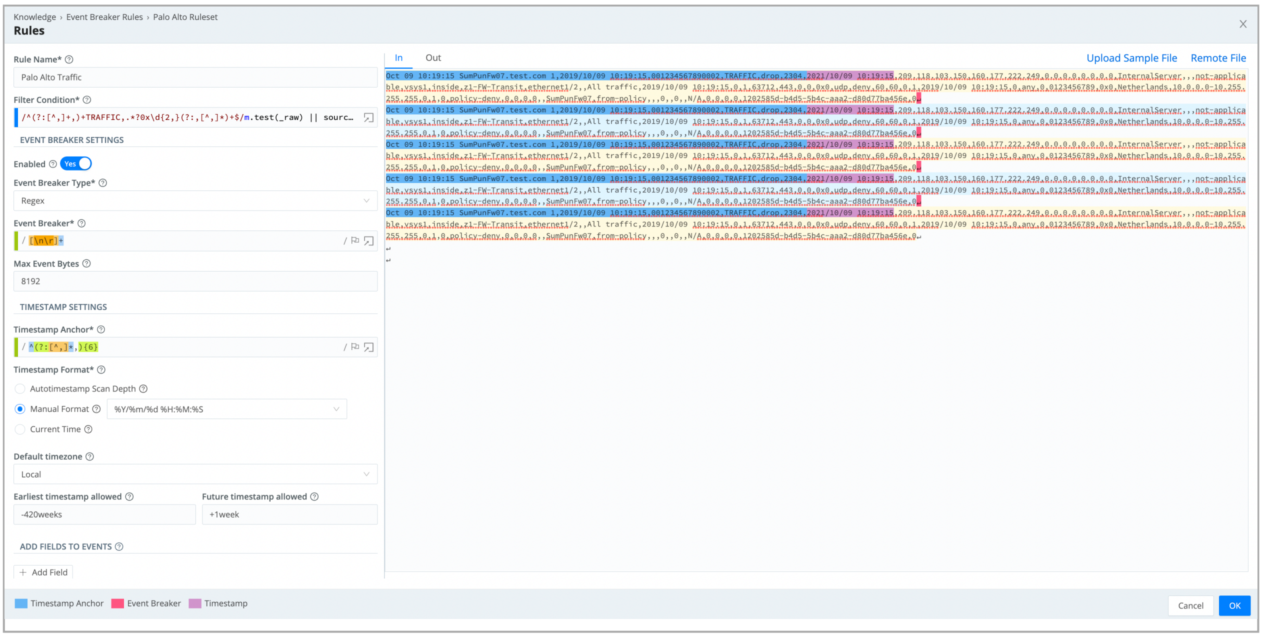
System Default Rule Filter Condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp Anchor: `/^/` Timestamp Format: `Auto:150` Default Timezone: `Local` Max Event Bytes: `51200`

Event Breaker buffer timeout: 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: %Y/%m/%d %H:%M:%S
○ Current Time

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

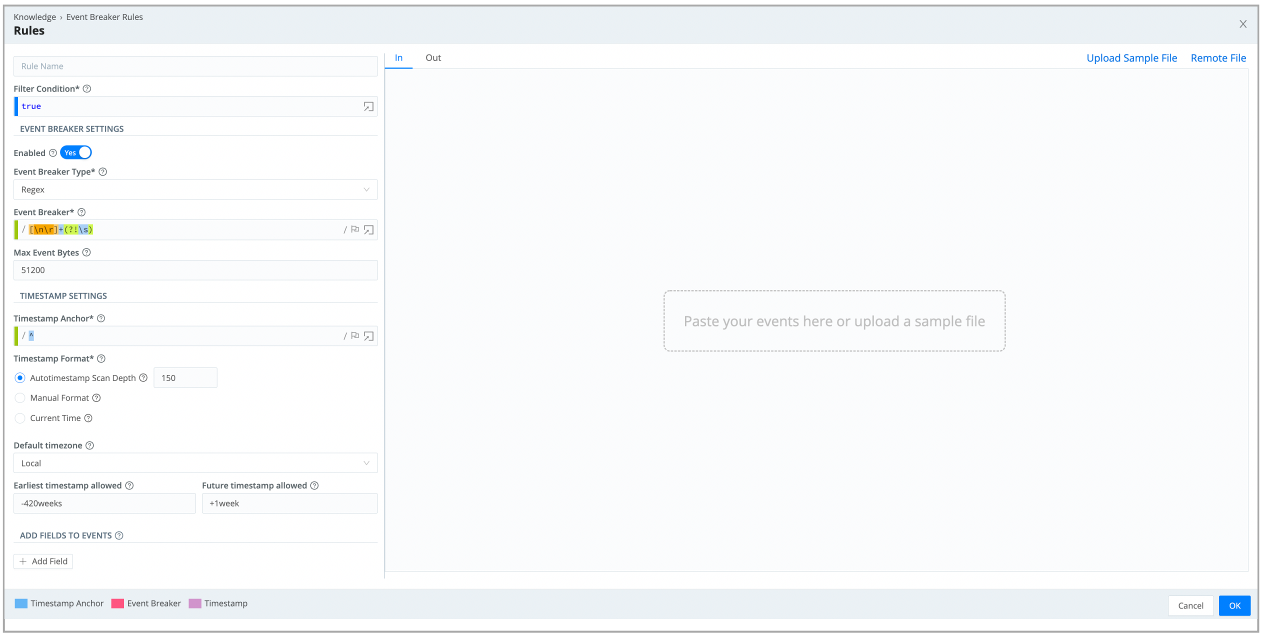
The system default rule functionally sits at the bottom of the ruleset/rule hierarchy (but is built-in and not displayed on the Event Breakers page), and goes into effect if there are no matching rules:

- 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 Rulesets** page (see screenshot above), click **+ Add New** 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 Rule** 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. Else, the next rule down the line is evaluated.

Event Breaker Type

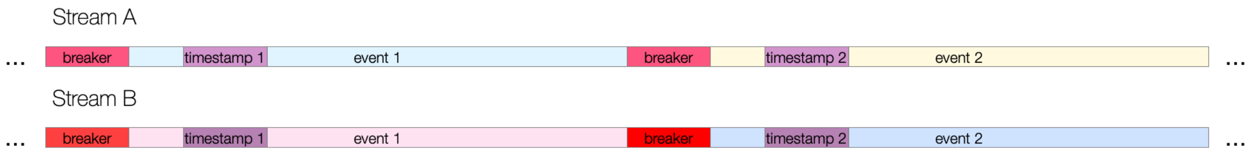
After a breaker pattern has been selected, it will apply on the stream **continuously**. See below for specific information on different Event Breaker Types.

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.
>
> 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.)

Event Breaker Types

Several types of Event Breaker can be applied to incoming data streams:

- Regex
- File Header
- JSON Array
- JSON New Line Delimited
- Timestamp
- CSV

Regex

The Regex breaker uses regular expressions to find breaking points in data streams.

After a breaker regex pattern has been selected, it will apply on the stream **continuously**. Breaking will occur at the beginning of the match, and the matched content will be consumed/thrown away. If necessary, you can use a positive lookahead regex to keep the content – e.g.: `(?=pattern)`

Capturing groups are **not allowed** to be used anywhere in the Event Breaker pattern, as they will further break the stream – which is often undesirable. Breaking will also occur if **Max Event Bytes** has been reached.

> The highest **Max Event Bytes** value that you can set is about 128 MB (`134217728` bytes). Events exceeding the maximum will be split into separate events, but left unbroken. Cribl Stream will set these events' `__isBroken` internal field to `false`.

Example

Break after a newline or carriage return, but only if followed by a timestamp pattern:
Event Breaker: `[\n\r]+(?=\d+-\d+-\d+\s\d+:\d+:\d+)`

```
--- input ---
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can
program the auxiliary card!
   Try to connect the FTP sensor, maybe it will connect the digital bus!
   Try to navigate the AGP panel, maybe it will quantify the mobile alarm!
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can
program the auxiliary card!
   Try to connect the FTP sensor, maybe it will connect the digital bus!
   Try to navigate the AGP panel, maybe it will quantify the mobile alarm!


--- output event 1 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then
you can program the auxiliary card! \n   Try to connect the FTP sensor, maybe it
will connect the digital bus!\n   Try to navigate the AGP panel, maybe it will
quantify the mobile alarm!",
  "_time": 1589920332
}

--- output event 2 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then
you can program the auxiliary card!\n   Try to connect the FTP sensor, maybe it will
connect the digital bus!\n   Try to navigate the AGP panel, maybe it will quantify
the mobile alarm!",
  "_time": 1589920332
}
```

File Header

You can use the File Header breaker to break files with headers, such as IIS or Bro logs. This type of breaker relies on a header section that lists field names. The header section is typically present at the top of the file, and can be single-line or greater.

After the file has been broken into events, fields will also be extracted, as follows:

- **Header Line**: Regex matching a file header line. For example, `^#`.
- **Field Delimiter**: Field delimiter regex. For example, `\s+`.
- **Field Regex**: Regex with one capturing group, capturing all the fields to be broken by field delimiter. For example, `^#[Ff]ields[:]?\s+(.*)`
- **Null Values**: Representation of a null value. Null fields are not added to events.
- **Clean Fields**: Whether to clean up field names by replacing non `[a-zA-Z0-9]` characters with `_`.

Example

Using the values above, let's see how this sample file breaks up:

```
--- input ---
#fields ts      uid     id.orig_h       id.orig_p       id.resp_h       id.resp_p
proto
#types  time    string  addr    port    addr    port    enum
1331904608.080000       -     192.168.204.59  137     192.168.204.255 137     udp
1331904609.190000       -     192.168.202.83  48516   192.168.207.4   53      udp


--- output event 1 ---
{
  "_raw": "1331904608.080000        -     192.168.204.59  137     192.168.204.255 137
udp",
  "ts": "1331904608.080000",
  "id_orig_h": "192.168.204.59",
  "id_orig_p": "137",
  "id_resp_h": "192.168.204.255",
  "id_resp_p": "137",
  "proto": "udp",
  "_time": 1331904608.08
}

--- output event 2 ---
{
  "_raw": "1331904609.190000        -     192.168.202.83  48516   192.168.207.4   53
udp",
  "ts": "1331904609.190000",
  "id_orig_h": "192.168.202.83",
  "id_orig_p": "48516",
  "id_resp_h": "192.168.207.4",
  "id_resp_p": "53",
  "proto": "udp",
  "_time": 1331904609.19
}
```

JSON Array

You can use the JSON Array to extract events from an array in a JSON document (e.g., an Amazon CloudTrail file).

- **Array Field**: Optional path to array in a JSON event with records to extract. For example, `Records`.
- **Timestamp Field**: Optional path to timestamp field in extracted events. For example, `eventTime` or `level1.level2.eventTime`.
- **JSON Extract Fields**: Enable this slider to auto-extract fields from JSON events. If disabled, only `_raw` and `time` will be defined on extracted events.
- **Timestamp Format**: If **JSON Extract Fields** is set to **No**, you **must** set this to **Autotimestamp** or **Current Time**. If **JSON Extract Fields** is set to **Yes**, you can select any option here.

Example

Using the values above, let's see how this sample file breaks up:

```
--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-
08T01:35:55Z","eventSource":"ec2.amazonaws.com","eventName":"DescribeVolumes",
"more_fields":"..."},
{"eventVersion":"1.05","eventTime":"2020-04-
08T01:35:56Z","eventSource":"ec2.amazonaws.com","eventName":"DescribeInstanceAttribute"
"more_fields":"..."}]}

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-
08T01:35:55Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"DescribeVolumes\",
\"more_fields\":\"...\"}",
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-
08T01:35:56Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"DescribeInstanceAt
\"more_fields\":\"...\"}",
  "_time": 1586309756,
  "cribl_breaker": "j-array"
}
```

JSON New Line Delimited

You can use the JSON New Line Delimited breaker to break and extract fields in newline-delimited JSON streams.

Example

Using default values, let's see how this sample stream breaks up:

```
--- input ---
{"time":"2020-05-
25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"info","message":"metric
sender","total":720,"dropped":0}
{"time":"2020-05-
25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"info","message":"metric
sender","total":720,"dropped":0}


--- output event 1 ---
{
  "_raw": "{\"time\":\"2020-05-
25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"clustercomm\",\"level\":\"info\",\"mess
sender\",\"total\":720,\"dropped\":0}",
  "time": "2020-05-25T18:00:54.201Z",
  "cid": "w1",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.201,
}

--- output event 2 ---
{
  "_raw": "{\"time\":\"2020-05-
25T18:00:54.246Z\",\"cid\":\"w0\",\"channel\":\"clustercomm\",\"level\":\"info\",\"mess
sender\",\"total\":720,\"dropped\":0}",
  "time": "2020-05-25T18:00:54.246Z",
  "cid": "w0",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.246,
}
```

Timestamp

You can use the Timestamp breaker to break events at the beginning of any line in which Cribl 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:

```
--- input ---
{"level":"debug","ts":"2021-02-
02T10:38:46.365Z","caller":"sdk/sync.go:42","msg":"Handle ENIConfig Add/Update: us-
west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42"}
{"level":"debug","ts":"2021-02-
02T10:38:56.365Z","caller":"sdk/sync.go:42","msg":"Handle ENIConfig Add/Update: us-
west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42"}


--- output event 1 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-
02T10:38:46.365Z\",\"caller\":\"sdk/sync.go:42\",\"msg\":\"Handle ENIConfig
Add/Update: us-west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42\"}",
  "_time": 1612262326.365
}

--- output event 2 ---
{
  "_raw": "{\"level\":\"debug\",\"ts\":\"2021-02-
02T10:38:56.365Z\",\"caller\":\"sdk/sync.go:42\",\"msg\":\"Handle ENIConfig
Add/Update: us-west-2a, [sg-426fdac8e5c22542], subnet-42658cf14a98b42\"}",
  "_time": 1612262336.365
}
```

CSV

The CSV breaker extracts fields in CSV streams that include a header line. Selecting this type exposes these extra fields:

- **Delimiter**: Delimiter character to use to split values. Defaults to: `,`

- **Quote Char**: Character used to quote literal values. Defaults to: `"`

- **Escape Char**: Character used to escape the quote character in field values. Defaults to: `"`

  Example: Using default values, let's see how this sample stream breaks up:

Example

Using default values, let's see how this sample stream breaks up:

```
--- input ---
time,host,source,model,serial,bytes_in,bytes_out,cpu
1611768713,"myHost1","anet","cisco","ASN4204269",11430,43322,0.78
1611768714,"myHost2","anet","cisco","ASN420423",345062,143433,0.28


--- output event 1 ---
{
  "_raw":
"\"1611768713\",\"myHost1\",\"anet\",\"cisco\",\"ASN4204269\",\"11430\",\"43322\",\"0.7
  "time": "1611768713",
  "host": "myHost1",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN4204269",
  "bytes_in": "11430",
  "bytes_out": "43322",
  "cpu": "0.78",
  "_time": 1611768713
}

--- output event 2 ---
{
  "_raw":
"\"1611768714\",\"myHost2\",\"anet\",\"cisco\",\"ASN420423\",\"345062\",\"143433\",\"0.
  "time": "1611768714",
  "host": "myHost2",
  "source": "anet",
  "model": "cisco",
  "serial": "ASN420423",
  "bytes_in": "345062",
  "bytes_out": "143433",
  "cpu": "0.28",
  "_time": 1611768714
}
```

> With **Type: CSV** selected, an Event Breaker will properly add quotes around all values, regardless of their initial state.

Cribl versus Custom Rulesets

Event Breaker rulesets shipped by Cribl will be listed under the **Cribl** tag, while user-built rulesets will be listed under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search.

Exporting and Importing Rulesets

You can export and import Custom (or Cribl) rulesets as JSON files. This facilitates sharing rulesets among Worker Groups or Cribl 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.

Troubleshooting

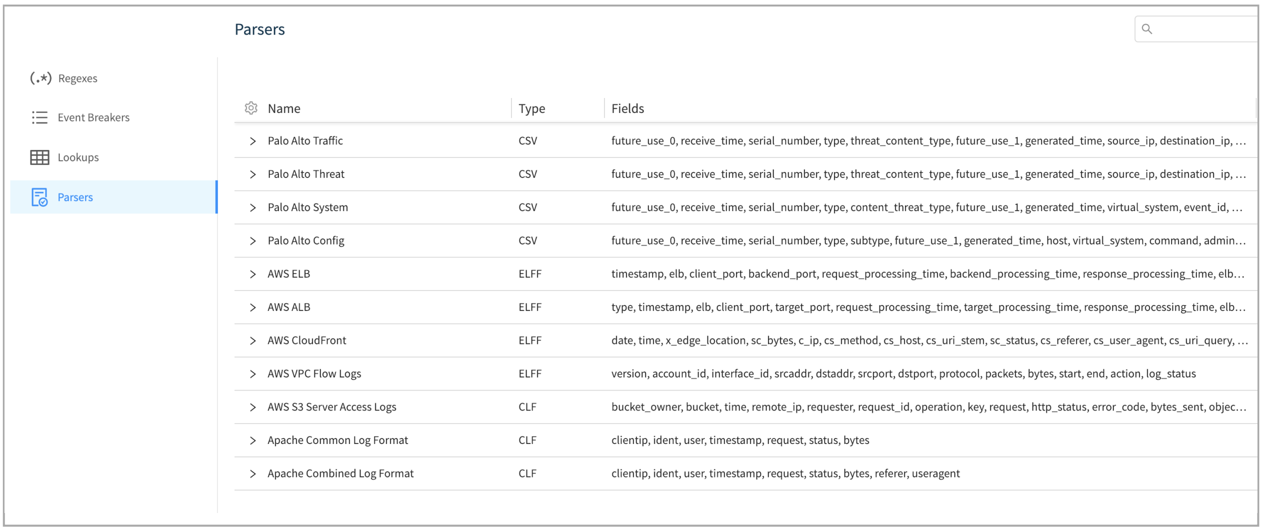
If you notice fragmented events, check whether Cribl Stream has added a `__timeoutFlush` internal field to them. This diagnostic field's presence indicates that the events were flushed because the Event Breaker buffer timed out while processing them. These timeouts can be due to large incoming events, backpressure, or other causes.

;

9.3. 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
Palo Alto Traffic	CSV	future_use_0, receive_time, serial_number, type, threat_content_type, future_use_1, generated_time, source_ip, destination_ip, …
Palo Alto Threat	CSV	future_use_0, receive_time, serial_number, type, threat_content_type, future_use_1, generated_time, source_ip, destination_ip, …
Palo Alto System	CSV	future_use_0, receive_time, serial_number, type, content_threat_type, future_use_1, generated_time, virtual_system, event_id, …
Palo Alto Config	CSV	future_use_0, receive_time, serial_number, type, subtype, future_use_1, generated_time, host, virtual_system, command, admin…
AWS ELB	ELFF	timestamp, elb, client_port, backend_port, request_processing_time, backend_processing_time, response_processing_time, elb…
AWS ALB	ELFF	type, timestamp, elb, client_port, target_port, request_processing_time, target_processing_time, response_processing_time, elb…
AWS CloudFront	ELFF	date, time, x_edge_location, sc_bytes, c_ip, cs_method, cs_host, cs_uri_stem, sc_status, cs_referer, cs_user_agent, cs_uri_query, …
AWS VPC Flow Logs	ELFF	version, account_id, interface_id, srcaddr, dstaddr, srcport, dstport, protocol, packets, bytes, start, end, action, log_status
AWS S3 Server Access Logs	CLF	bucket_owner, bucket, time, remote_ip, requester, request_id, operation, key, request, http_status, error_code, bytes_sent, objec…
Apache Common Log Format	CLF	clientip, ident, user, timestamp, request, status, bytes
Apache Combined Log Format	CLF	clientip, ident, user, timestamp, request, status, bytes, referer, useragent

Parsers Library

Parsers can be used to **extract** or **reserialize** events. See Parser Function page for examples.

Supported Parser Types:

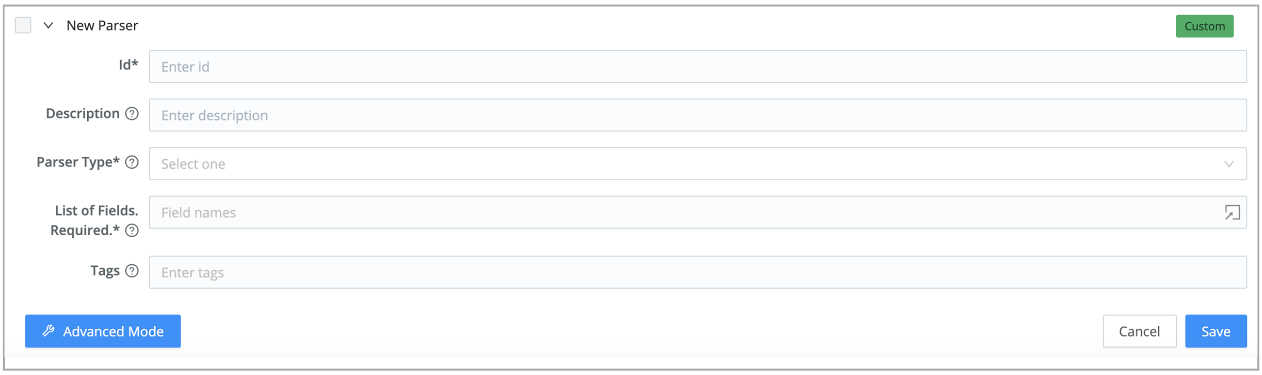
- CSV – Parse and reserialize comma-separated values.
- ELFF – Parse and reserialize events in Extended Log File Format.
- CLF – Parse and reserialize events in Common Log Format.

Creating a Parser

To create a parser, follow these steps:

1. Go to **Knowledge > Parsers** and click **Add New**.
2. Enter a unique **ID**.
3. Optionally, enter a **Description**.

4. Select a **Parser type** (see the supported types above).
5. Enter the **List of fields** expected to be extracted, in order. Click this field's Maximize icon (far right) if you'd like to open a modal where you can work with sample data and iterate on results.
6. Optionally, enter any desired **Tags**.



□ New Parser

Custom

Id* Enter id

Description Enter description

Parser Type* Select one

List of Fields. Required.* Field names

Tags Enter tags

Adding a new parser

;

9.4. 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`

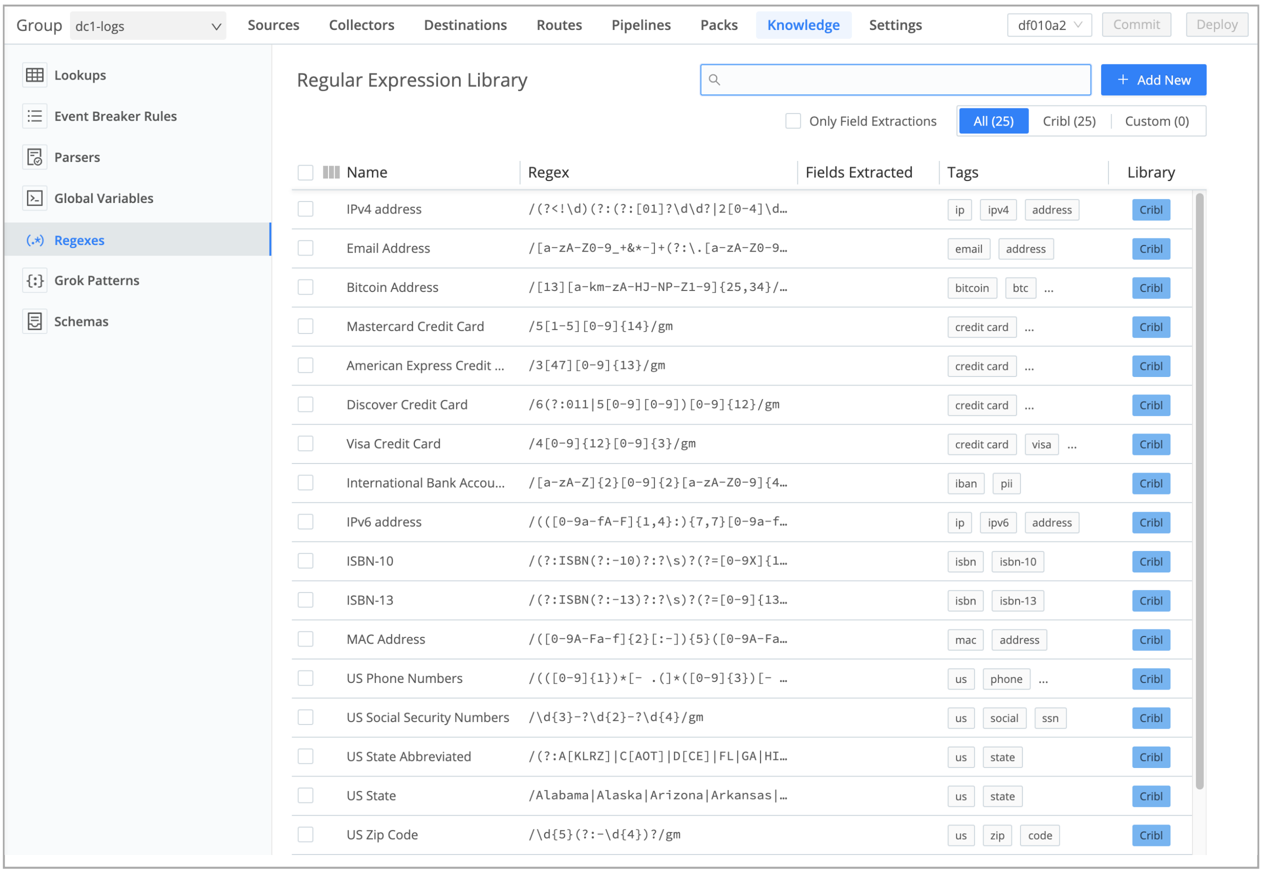
- **Sample Eval Function:** `storage = C.vars.convertBytes(size)`
  Note the use of `bytes` here as an argument.

;

9.5. 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** > **Regex Library** .



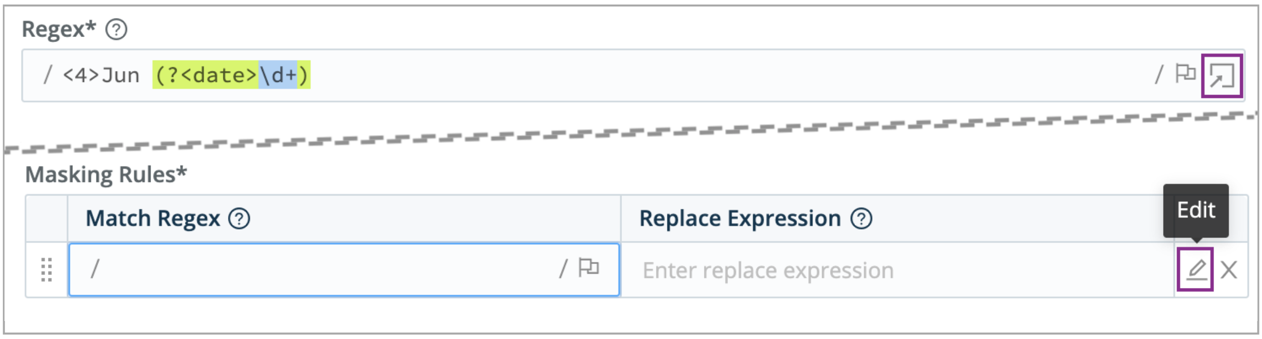
Regular Expression Library

<table>
<tr><th>Name</th><th>Regex</th><th>Fields Extracted</th><th>Tags</th><th>Library</th></tr>
<tr><td>IPv4 address</td><td>/(?&lt;!\d)(?:(?:[01]?\d\d?|2[0-4]\d…</td><td></td><td>ip
ipv4
address</td><td>Cribl</td></tr>
<tr><td>Email Address</td><td>/[a-zA-Z0-9_+&amp;*-]+(?:\.[a-zA-Z0-9…</td><td></td><td>email
address</td><td>Cribl</td></tr>
<tr><td>Bitcoin Address</td><td>/[13][a-km-zA-HJ-NP-Z1-9]{25,34}/…</td><td></td><td>bitcoin
btc
...</td><td>Cribl</td></tr>
<tr><td>Mastercard Credit Card</td><td>/5[1-5][0-9]{14}/gm</td><td></td><td>credit card
...</td><td>Cribl</td></tr>
<tr><td>American Express Credit ...</td><td>/3[47][0-9]{13}/gm</td><td></td><td>credit card
...</td><td>Cribl</td></tr>
<tr><td>Discover Credit Card</td><td>/6(?:011|5[0-9][0-9])[0-9]{12}/gm</td><td></td><td>credit card
...</td><td>Cribl</td></tr>
<tr><td>Visa Credit Card</td><td>/4[0-9]{12}[0-9]{3}/gm</td><td></td><td>credit card
visa
...</td><td>Cribl</td></tr>
<tr><td>International Bank Accou...</td><td>/[a-zA-Z]{2}[0-9]{2}[a-zA-Z0-9]{4…</td><td></td><td>iban
pii</td><td>Cribl</td></tr>
<tr><td>IPv6 address</td><td>/(([0-9a-fA-F]{1,4}:){7,7}[0-9a-f…</td><td></td><td>ip
ipv6
address</td><td>Cribl</td></tr>
<tr><td>ISBN-10</td><td>/(?:ISBN(?:-10)?:?\s)?(?=[0-9X]{1…</td><td></td><td>isbn
isbn-10</td><td>Cribl</td></tr>
<tr><td>ISBN-13</td><td>/(?:ISBN(?:-13)?:?\s)?(?=[0-9]{13…</td><td></td><td>isbn
isbn-13</td><td>Cribl</td></tr>
<tr><td>MAC Address</td><td>/([0-9A-Fa-f]{2}[:-]){5}([0-9A-Fa…</td><td></td><td>mac
address</td><td>Cribl</td></tr>
<tr><td>US Phone Numbers</td><td>/(([0-9]{1})*[- .(]*([0-9]{3})[- …</td><td></td><td>us
phone
...</td><td>Cribl</td></tr>
<tr><td>US Social Security Numbers</td><td>/\d{3}-?\d{2}-?\d{4}/gm</td><td></td><td>us
social
ssn</td><td>Cribl</td></tr>
<tr><td>US State Abbreviated</td><td>/(?:A[KLRZ]|C[AOT]|D[CE]|FL|GA|HI…</td><td></td><td>us
state</td><td>Cribl</td></tr>
<tr><td>US State</td><td>/Alabama|Alaska|Arizona|Arkansas|…</td><td></td><td>us
state</td><td>Cribl</td></tr>
<tr><td>US Zip Code</td><td>/\d{5}(?:-\d{4})?/gm</td><td></td><td>us
zip
code</td><td>Cribl</td></tr>
</table>

Regular Expression Library

Using Library Patterns

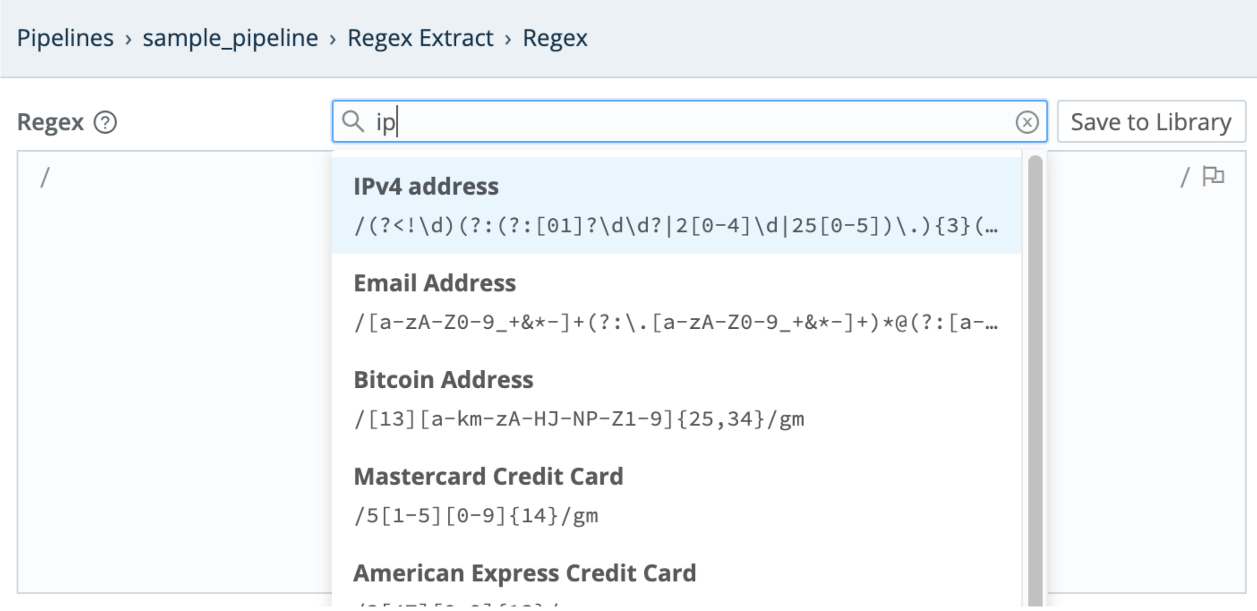
As of this version, the Library contains 25 patterns shipped by Cribl Stream. To insert a pattern into a Function's regex field, first click the pop-out or Edit icon beside that field.



```
/ <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.



ip

IPv4 address
`/(?<!\d)(?:(?:[01]?\d\d?|2[0-4]\d|25[0-5])\.){3}(…`

Email Address
`/[a-zA-Z0-9_+&*-]+(?:\.[a-zA-Z0-9_+&*-]+)*@(?:[a-…`

Bitcoin Address
`/[13][a-km-zA-HJ-NP-Z1-9]{25,34}/gm`

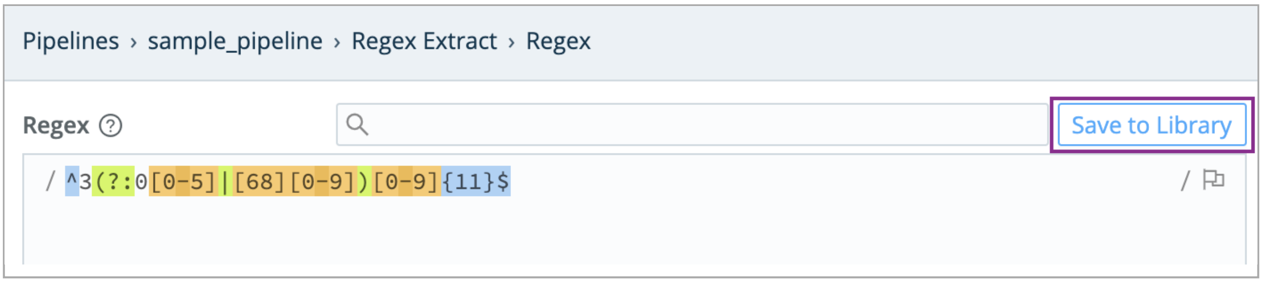
Mastercard Credit Card
`/5[1-5][0-9]{14}/gm`

American Express Credit Card

Inserting a pattern from the Regex Library

Adding Patterns to the Library

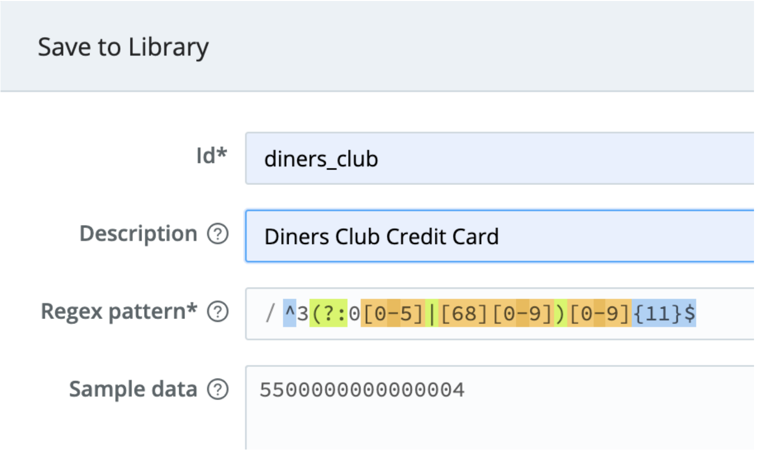
You can also add new, custom patterns to the Library. In the same modal, once you've built your pattern, click the **Save to Library** button.



```
/ ^3(?:0[0-5]|[68][0-9])[0-9]{11}$
```

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.

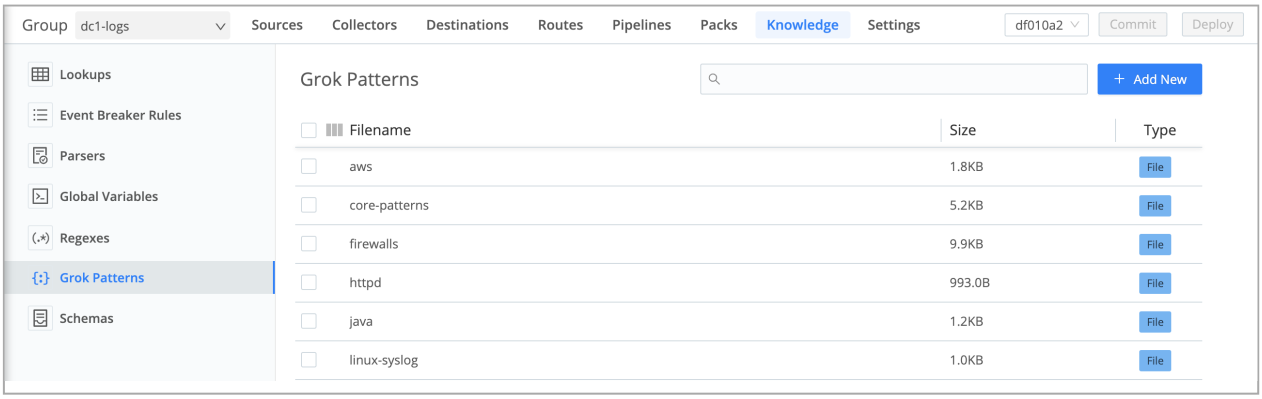
In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search. For example, if a Cribl-provided pattern and a Custom one are both named `ipv4`, the one from Cribl will not be displayed or delivered as a search result.

;

9.6. 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.



Filename	Size	Type
aws	1.8KB	File
core-patterns	5.2KB	File
firewalls	9.9KB	File
httpd	993.0B	File
java	1.2KB	File
linux-syslog	1.0KB	File

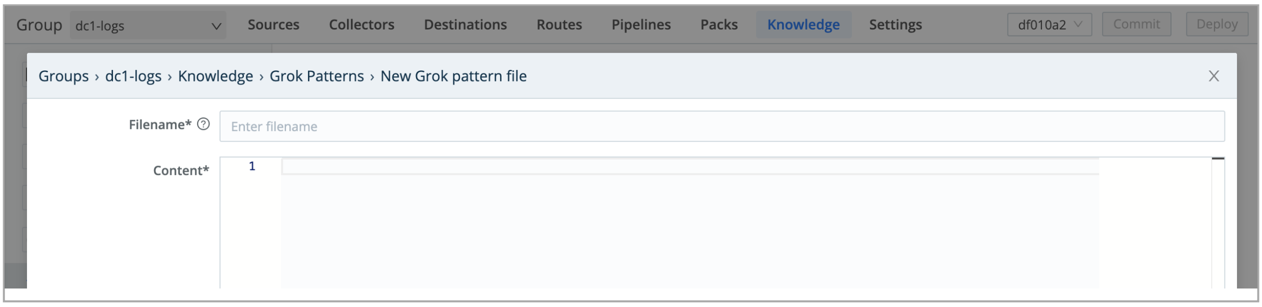
Grok Patterns Library

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 New**. In the resulting **Create Grok Patterns** modal, assign a unique **Filename**, populate the file with patterns, then click **Save**.



Adding Grok patterns

> Pattern files reside in: `$CRIBL_HOME/(default|local)/cribl/grok-patterns/`

Using Grok Patterns

In the current Cribl 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).

;

9.7. Schemas Library

Cribl Stream supports two kinds of schemas:

- JSON schemas for validating JSON events. Manage these in the UI at **Knowledge** > **Schemas**.
- Parquet schemas for writing data from a Cribl Stream Destination to Parquet files. Manage these in the UI at **Knowledge** > **Parquet Schemas**.

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 **+ Add New**. Enter the following:

- ID: `schema1.`
- Description: (Enter your own description here.)
- Schema: Paste the following schema.

JSON Schema - Sample

```
{
  "$id": "https://example.com/person.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Person",
  "type": "object",
  "required": ["firstName", "lastName", "age"],
  "properties": {
    "firstName": {
      "type": "string",
      "description": "The person's first name."
    },
    "lastName": {
      "type": "string",
      "description": "The person's last name."
    },
    "age": {
      "description": "Age in years which must be equal to or greater than zero.",
      "type": "integer",
      "minimum": 0,
      "maximum": 42
    }
  }
}
```

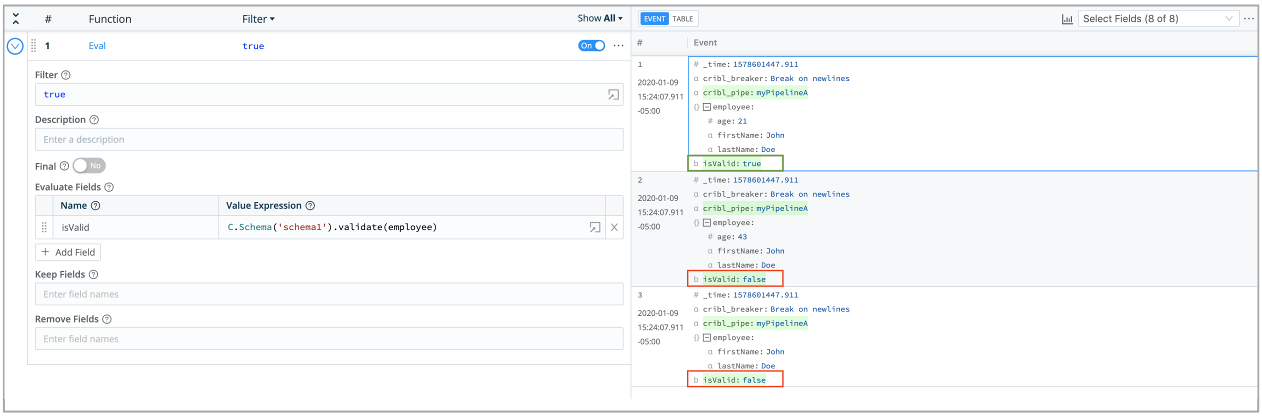
Assume that events look like this:

```
{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}
```

To validate whether the `employee` field is valid per `schema1`, we can use the following: `C.Schema('schema1').validate(employee)`

Results:

- First event is **valid**.
- Second event is **not valid** because `age` is greater than the maximum of `42`.
- Third event is **not valid** because `age` is missing.



#	Function	Filter	
1	Eval	true	On

Filter: `true`

Description: Enter a description

Final: No

Evaluate Fields

Name	Value Expression
isValid	`C.Schema('schema1').validate(employee)`

Keep Fields: Enter field names

Remove Fields: Enter field names

#	Event
1 2020-01-09 15:24:07.911 -05:00	# _time: 1578601447.911 α cribl_breaker: Break on newlines α cribl_pipe: myPipelineA {} employee: # age: 21 α firstName: John α lastName: Doe b isValid: true
2 2020-01-09 15:24:07.911 -05:00	# _time: 1578601447.911 α cribl_breaker: Break on newlines α cribl_pipe: myPipelineA {} employee: # age: 43 α firstName: John α lastName: Doe b isValid: false
3 2020-01-09 15:24:07.911 -05:00	# _time: 1578601447.911 α cribl_breaker: Break on newlines α cribl_pipe: myPipelineA {} employee: α firstName: John α lastName: Doe b isValid: false

Schema validation results for the above events

Parquet Schemas

Destinations whose **General Settings** > **Data format** drop-down includes a `Parquet` option can write out data as files in the Apache Parquet columnar storage format.

Before configuring a Destination for Parquet output, you should add an existing Parquet schema, or create a new Parquet schema, that suits the data you're working with. You do not need to start from scratch: Cribl provides sample Parquet schemas for you to clone and then customize as needed.

From the top menu, select **Knowledge**, then select **Parquet Schemas** from the left nav.

If you're adding a Parquet schema:

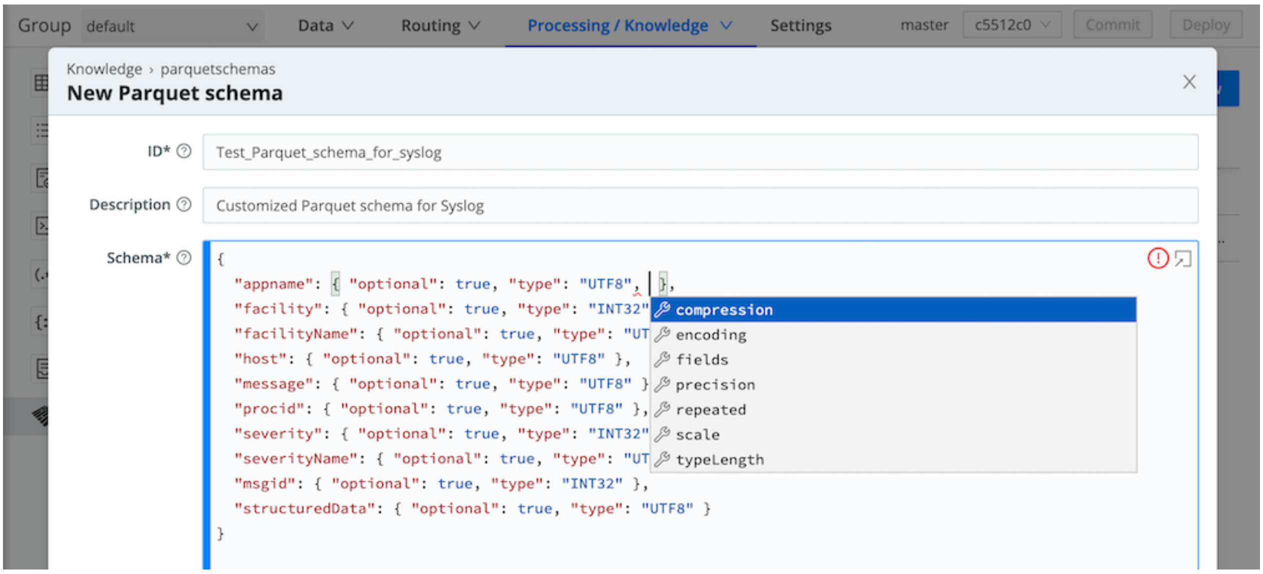
- Click **+ Add New** to open the **New Parquet schema** modal.

If you're creating a Parquet 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.

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 3.5.3 and above.

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, except `LIST` and `MAP`.

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.

;

10. Reference

10.1. 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.

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.<organization_name>.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 token. You obtain the Bearer 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.

Cribl.Cloud

On Cribl.Cloud, you cannot get the Bearer token directly. Instead, get it from the API Reference:

1. Select global ⚙ **Settings** (lower left) > **API Reference**.
2. Click on any `GET` endpoint that requires no parameters.
3. In that endpoint's open accordion, click **Try it out**, then **Execute**.
4. From the resulting `curl` response, copy the token value after `Authorization: Bearer`.
5. Include that value in the Authorization header, like this:
   `Bearer <value>`

On-Prem

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 enable the **Allow local auth** setting. in order for local users to authenticate using the API.

Request:

```
POST https://<cribl-stream-leader>:9000/api/v1/auth/login
```

Payload:

```
{
    "username": "<username>",
    "password": "<password>"
}
```

Example request using `curl`:

```
curl -X POST https://<cribl-stream-leader>:9000/api/v1/auth/login -d '{"username":"
<username>","password":"<password>"}'
```

You'll get a JSON object as the response, e.g.:

```
{
    "token": "<JWT_Token>",
    "forcePasswordChange": false
}
```

Use this Bearer token in all subsequent requests. Include it in the Authorization header, like this:

```
Bearer <JWT_token>
```

Update Basic Configurations

You can use the API to programmatically update the configuration of any object type that the API supports (e.g., Sources and Destinations).

Example: Periodically rotate S3 keys on a preconfigured S3/MinIO destination.

1. Send a `GET` request to the `/outputs` endpoint to retrieve the definition for a Destination (in this case, `MinIO`). The response provides the definition in its payload, as a JSON object.

   Example request:

   ```
   curl -X GET "<url>/api/v1/system/inputs/<output id>" -H "accept:
   application/json" -H "Authorization: Bearer <token>"
   ```

   Example response:

```
{
  "systemFields": [
    "cribl_pipe"
  ],
  "signatureVersion": "v4",
  "objectACL": "private",
  "partitionExpr": "`${host}/${sourcetype}`",
  "format": "json",
  "baseFileName": "CriblOut",
  "compress": "none",
  "maxFileSizeMB": 32,
  "maxFileOpenTimeSec": 300,
  "maxFileIdleTimeSec": 30,
  "maxOpenFiles": 100,
  "onBackpressure": "block",
  "id": "minio",
  "type": "minio",
  "endpoint": "http://minio:9090",
  "bucket": "test",
  "destPath": "keyprefix",
  "stagePath": "tmp",
  "awsApiKey": "minioadmin",
  "awsSecretKey": "minioadmin"
}
```

2. Edit the definition so that when you send it back in a `PATCH` request, it updates the desired Destination's S3 keys.

   - Edit the value of the `id` field to be the ID of the specific Destination whose keys you want to rotate, e.g., `minio_042`.
   - Edit the values of the S3 key fields, e.g., `"awsApiKey": "minioadmin_new_api_key"` and `"awsSecretKey": "minioadmin_new_secret_key"`.

3. Send the edited definition as the payload of a `PATCH` request to the `/outputs` enpdoint. 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:

```
curl -X PUT "<url>/api/v1/m/default/system/lookups?filename=example.csv" \
-H "Authorization: Bearer <token>" -H 'Content-Type: text/csv' \
--data-binary '@/path/to/your/example.csv'
```

You will receive a JSON object response similar to the following example:

```
{"filename":"example.csv.random.tmp","rows":100,"size":200}
```

2. Send a `POST` request referencing the Lookup file to the `/system/lookups` endpoint. Replace the filename with the response from the previous **PUT** request. This both creates the Lookup and moves the Lookup file to its final destination. Example:

   ```
   curl -X POST "<url>/api/v1/m/default/system/lookups" \
   -H "accept: application/json" -H "Authorization: Bearer <token>" \
   -H "Content-Type: application/json" \
   -d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
   ```

   Example response:

   ```
   {
     "items": [
       {
         "id": "example.csv",
         "size": 200
       }
     ],
     "count": 1
   }
   ```

3. If replacing an existing lookup, send a `PATCH` request referencing the existing filename in the URL and the body. Example:

   ```
   curl -X PATCH "<url>/api/v1/m/default/system/lookups/example.csv" \
   -H "accept: application/json" -H "Authorization: Bearer <token>" \
   -H "Content-Type: application/json" \
   -d '{"id":"example.csv","fileInfo":{"filename":"example.csv.random.tmp"}}'
   ```

   Example response:

   ```
   {
     "items": [
       {
         "id": "example.csv",
         "size": 200
       }
     ],
     "count": 1
   }
   ```

Distributed Upload

In a distributed environment, for lookup files of manageable size: Cribl recommends uploading the Lookup file only to the Leader Node, and then making a selective commit and deploy with only those Lookup file changes. The Leader then notifies 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 .

;

10.2. 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

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

TRUTHY	FALSY
`true`	`false`
`42`	`null`
`-42`	`undefined`
`3.14`	`0`
`"foo"`	`NaN`
`Infinity`	`''`
`-Infinity`	`""`

Value Expressions

Value expressions are typically used in Functions to assign a value – for example, to a new field. For example:

- `Math.floor(_time/3600)`
- `source.replace(/.{3}/, 'XXX')`

Best Practices for Creating Predictable Expressions

- In a value expression, ensure that the source variable is not `null`, `undefined`, or `empty`. For example, assume you want to have a field called `len`, to be assigned the length of a second field called `employeeID`. But you're not sure if `employeeID` exists. Instead of `employeeID.length`, you can use a safer shorthand, such as: `(employeeID || '').length`.

- If a field does not exist (undefined), and you're doing a comparison with its properties, then the boolean expression will **always** evaluate to false. For example, if `employeeID` is undefined, then both of these expressions will evaluate to false: `employeeID.length > 10` , and `employeeID.length < 10` .

- `==` means "equal to," while `===` means "equal value **and** equal type." For example, `5 == 5` and `5 == "5"` each evaluate to **true**, while `5 === "5"` evaluates to **false**.

- A ternary operator is a very powerful way to create conditional values. For example, if you wanted to assign either `minor` or `adult` to a field `groupAge`, based on the value of `age`, you could do: `(age >= 18) ? 'adult' : 'minor'`.

Fields with Non-Alphanumeric Characters

If there are fields whose names include non-alphanumeric characters – e.g., `@timestamp` or `user-agent` or `kubernetes.namespace_name` – you can access them using `__e['<field-name-here>']`. (Note the single quotes.) More details here.

In any other place where the field is referenced – e.g., in the Eval function's field names – you should use a single-quoted literal, of the form: `'<field-name-here>'`.

Wildcard Lists

Wildcard Lists are used throughout the product, especially in various Functions, such as Eval, Mask, Publish Metrics, Parser, etc.

Wildcard Lists, as their name implies, accept strings with asterisks (`*`) to represent one or more terms. They also accept strings that start with an exclamation mark (`!`) to **negate** one or more terms.

Wildcard Lists are order-sensitive **only** when negated terms are used. This allows for implementing any combination of allowlists and blocklists.

For example:

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**.

> 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*`.

;

10.3. Cribl Expressions

Native Cribl Stream function methods can be found under `C.*`, and can be invoked from any Function that allows for expression evaluations. For example, to create a field that is the SHA1 of a another field's value, you can use the Eval Function with this **Evaluate Fields** pair:

NAME	VALUE EXPRESSION
myNewField	`C.Mask.sha1(myOtherField)`

> Where fields' names contain special characters, you can reference them using the `__e['<field-name-here>']` convention. For details, see Fields with Non-Alphanumeric Characters.

C.Crypto – Data Encryption and Decryption Functions

`C.Crypto.decrypt`
(method) `Crypto.decrypt(value: string, escape: boolean, escapeSeq: string): string`
Decrypt all occurrences of ciphers in the given value. Instances that cannot be decrypted (for any reason) are left intact.
@param – `value` – String in which to look for ciphers.
@param – `escape` – Boolean, defaults to `false`. Set to `true` to escape double quotes in output after decryption. (E.g., for data encrypted in Splunk.)
@param – `escapeSeq` – String used to escape double quotes. The default `'"'` escapes CSV output.
@returns – `value` with ciphers decrypted.

`C.Crypto.encrypt`
(method) `Crypto.encrypt(value: any, keyclass: number, keyId?: string, defaultVal?: string): string`
Encrypt the given value with the `keyId`, or with a `keyId` picked up automatically based on `keyclass`.
@param {string | Buffer} `value` – what to encrypt.
@param – `keyclass` – if `keyId` isn't specified, pick one at the given `keyclass`.
@param – `keyId` - encryption keyId, takes precedence over `keyclass`.
@param – `defaultVal` – what to return if encryption fails for any reason; if unspecified, the original value is returned.
@returns – if encryption succeeds, the encrypted value; otherwise, `defaultVal` if specified; otherwise, `value`.

`C.Crypto.createHmac`

(method) `Crypto.createHmac(value: string | Buffer, secret: string, algorithm: string = 'sha256', outputFormat: 'base64' | 'hex' | 'latin1' = 'hex'): string`

Generates an HMAC that can be added to events, or can be used to validate events that contain an HMAC. (Available in Cribl Stream v.3.1.2+.)

@param – `value` – The data to encrypt, as a string. (When the `outputFormat` is invalid or undefined, this parameter is returned as the digest, via a Buffer.)

@param} – `secret` – The secret key used to generate the MAC, as a string.

@param – `algorithm` – The hash algorithm used to generate the MAC, as a string. Defaults to `'sha256'`. Run `openssl list -digest-algorithms` to see the list of available algorithms.

@param – `outputFormat` – One of `'base64'`, `'hex'`, or `'latin1'`. Defaults to `'hex'`.

@returns – The calculated HMAC digest on success; otherwise, `value`.

C.Decode – Data Decoding Functions

`C.Decode.base64`

(method) `Decode.base64(val: string, resultEnc?: string): any`

Performs base64 decoding of the given string. Returns a string or Buffer, depending on the `resultEnc` value, which defaults to `'utf8'`.

@param – `val` – value to base64-decode.

@param – `resultEnc` – encoding to use to convert the binary data to a string. Defaults to `'utf8'`. Use `'utf8-valid'` to validate that result is valid UTF8; use `'buffer'` if you need the binary data in a Buffer.

`C.Decode.gzip`

(method) `Decode.gzip(value: any, encoding?: string): string`

Gunzip the supplied value.

@param – `value` – the value to gunzip.

@param – `encoding` – encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`. Default is `'base64'`. If data is received as Buffer (from gzip with encoding: `'none'`), decoding is skipped.

`C.Decode.hex`

(method) `Decode.hex(val: string): number`

Performs hex to number conversion. (Returns `NaN` if value cannot be converted to a number.)

@param – `val` – hex string to parse to a number (e.g., "0xcafe").

`C.Decode.uri`

(method) `Decode.uri(val: string): string`

Performs URI-decoding of the given string.

@param – `val` – value to URI-decode.

C.Encode – Data Encoding Functions

`C.Encode.base64`
(method) `Encode.base64(val: any, trimTrailEq?: boolean): string`
Returns a base64 representation of the given string or Buffer.
@param – `val` – value to base64-encode.
@param – `trimTrailEq` – whether to trim any trailing `=`.

`C.Encode.gzip`
(method) `Encode.gzip(value: string, encoding?: string): any`
Gzip, and optionally base64-encode, the supplied value.
@param – `value` – the value to gzip.
@param – `encoding` – encoding of `value`, for example: `'base64'`, `'hex'`, `'utf-8'`, `'binary'`, `'none'`. Default is `'base64'`. If `'none'` is specified, data will be returned as a Buffer.

`C.Encode.hex`
(method) `Encode.hex(val: string | number): string`
Rounds the number to an integer and returns its hex representation (lowercase). If a string is provided, it will be parsed into a number or `NaN`.
@param – `val` – value to convert to hex.

`C.Encode.uri`
(method) `Encode.uri(val: string): string`
Returns the URI-encoded representation of the given string.
@param – `val` – value to uri encode.

C.env – Environment

`C.env`
(property) `env: {[key: string]: string;}`
Returns an object containing Cribl Stream's environment variables.

C.Lookup – Inline Lookup Functions

`C.Lookup` – Exact Lookup
(property) `Lookup: (file: string, primaryKey?: string, otherFields?: string[], ignoreCase?: boolean) => InlineLookup`
Returns an instance of a lookup to use inline.

Example invocation, where `host` is the name of the primary key field:
`C.Lookup('lookup_name.csv', 'IP_field_name_in_lookup_file').match(host)`

Expanded example, where the quoted `'event_field_or_string_to_match'` could be a string to match in the primary key field:
`C.Lookup('name_of_lookup_file.csv', 'field_in_csv_to_match').match('event_field_or_string_to_match', 'field_in_csv_to_output')`

> `C.Lookup` can load lookup files of up to 10 MB.
>
> All inputs to Lookup functions' `match()` method must be strings. If your lookup file contains numeric fields, convert them to strings, e.g.: `.match(String(<fieldname>)`.
>
> The optional `otherFields[]` argument shown in `C.Lookup()` signatures and examples is reserved for future use, and not currently implemented.

`C.LookupCIDR` – CIDR Lookup
(property) `LookupCIDR: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`
Returns an instance of a CIDR lookup to use inline.

`C.LookupIgnoreCase` – Case-insensitive Lookup
(property) `LookupIgnoreCase: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`
Returns an instance of a lookup (ignoring case) to use inline. Works identically to `C.Lookup`, except ignores the case of lookup values. (Equivalent to calling `C.Lookup` with its fourth `ignoreCase?` parameter set to `true`).

`C.[LookupRegex](http://google.com)` – Regex Lookup
(property) `LookupRegex: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup`
Returns an instance of a Regex lookup to use inline.

(method) `InlineLookup.match(value: string, fieldToReturn?: string): any`
@param – `value` – the value to look up.
@param – `fieldToReturn` – name of the lookup file > field to return.

E.g., `C.Lookup('lookup-exact.csv', 'foo').match('abc', 'bar')`
Return the value of field **bar** in the lookup table if field **foo** matches `abc`.

Example 1: `C.LookupCIDR('lookup-cidr.csv', 'foo').match('192.168.1.1', 'bar')`
Return the value of field **bar** in the lookup table if the CIDR range in **foo** includes `192.168.1.1`.

Example 2: `C.LookupCIDR('lookup-cidr.csv', 'cidr').match(hostIP, 'location')` Return the value of column **location** in the lookup table if the **location** includes `hostIP

Example 3: `C.LookupRegex('lookup-regex.csv', 'foo').match('manchester', 'bar')`
Return the value of field **bar** in the lookup table if the regex in **foo** matches the string `manchester`.

> With `C.LookupRegex`, ensure that your lookup file contains no empty lines – not even at the bottom. Any empty row will cause `C.LookupRegex().match()` to always return `true`.

C.Mask – Data Masking Functions

`C.Mask.CC` (method) `Mask.CC(value: string, unmasked?: number, maskChar?: string): string`
Check whether a value could be a valid credit card number, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`.
@param – `value` – a string whose digits to mask IFF it could be a valid credit card number.
@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.
@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.IMEI` (method) `Mask.IMEI(value: string, unmasked?: number, maskChar?: string): string`
Check whether a value could be a valid IMEI number, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`.
@param – `value` – a string whose digits to mask IFF it could be a valid IMEI number.
@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.
@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.isCC`
(method) `Mask.isCC(value: string): boolean`
Checks whether the given value could be a valid credit card number, by computing the string's Lunh's checksum modulo 10 == `0`.
@param – `value` – a string to check for being a valid credit card number.

`C.Mask.isIMEI`
(method) `Mask.isIMEI(value: string): boolean`
Checks whether the given value could be a valid IMEI number, by computing the string's Lunh's checksum modulo 10 == `0`.
@param – `value` – a string to check for being a valid IMEI number

`C.Mask.luhn`

(method) `Mask.luhn(value: string, unmasked?: number, maskChar?: string): string`

Check that value Lunh's checksum mod 10 is `0`, and mask a subset of the value. By default, all digits except the last 4 will be replaced with `X`. If the value's Lunh's checksum mod 10 is not `0`, then the value is returned unmodified.

@param – `value` – a string whose digits to mask IFF the value's Lunh's checksum mod 10 is `0`.

@param – `unmasked` – number of digits to leave unmasked: positive for left, negative for right, `0` for none.

@param – `maskChar` – a string/char to replace a digit with.

`C.Mask.LUHN_SUB` (property) `Mask.LUHN_SUB: any`

`C.Mask.luhnChecksum`

(method) `Mask.luhnChecksum(value: string, mod?: number): number`

Generates the Luhn checksum (used to validate certain credit card numbers, IMEIs, etc.). By default, the mod 10 of the checksum is returned. Pass mod = `0` to get the actual checksum.

@param – `value` – a string whose digits you want to perform the Lunh checksum on.

@param – `mod` – return checksum modulo this number. If `0`, skip modulo. Default is `10`.

`C.Mask.md5`

(method) `Mask.md5(value: string, len?: string | number): string`

Generate MD5 hash of a given value.

@param – `value` – compute the hash of this.

@param – `len` – length of hash to return: 0 for full hash, a +number for left or a -number for right substring. If a string is passed it's length will be used.

`C.Mask.random`

(method) `Mask.random(len?: string | number): string`

Generates a random alphanumeric string.

@param – `len` – a number indicating the length of the result; or, if a string, use its length.

`C.Mask.REDACTED`

(property) `Mask.REDACTED: string`

The literal `'REDACTED'`.

`C.Mask.repeat`

(method) `Mask.repeat(len?: string | number, char?: string): string`

Generates a repeating char/string pattern, e.g., `XXXX`.

@param – `len` – a number indicating the length of the result; or, if a string, use its length.

@param – `char` – pattern to repeat `len` times.

`C.Mask.sha1` (method) `Mask.sha1(value: string, len?: string | number): string` Generate SHA1 hash of given value.

@param – `value` - compute the hash of this.
@param – `len` - length of hash to return: `0` for full hash, a +number for left, or a -number for right.
substring. If a string is passed, its length will be used

C.Misc – Miscellaneous Utility Functions

`C.Misc.zip()`
(method) `Misc.zip(keys: string[], values: any[], dest?: any): any`
Set the given keys to the corresponding values on the given `dest` object. If `dest` is not provided, a new object will be constructed.
@param – `keys` – field names corresponding to keys.
@param – `values` – values corresponding to values.
@param – `dest` – object on which to set field values.
@returns – object on which the fields were set.

E.g., `people = C.Misc.zip(titles, names)`
Sample data: `titles=['ceo', 'svp', 'vp']`, `names=['foo', 'bar', 'baz']`
Create an object called `people`, with key names from elements in `titles`, and with corresponding values from elements in `names`.
Result: `"people": {"ceo": "foo", "svp": "bar", "vp": "baz"}`

C.Net – Network Functions

`C.Net.cidrMatch()`
(method) `Net.cidrMatch(cidrIpRange: string, ipAddress: string): boolean`
Determines whether the supplied IPv4 `ipAddress` is inside the range of addresses identified by `cidrIpRange`. For example: `C.Net.cidrMatch ('10.0.0.0/24', '10.0.0.100')` returns `true.`
@param – `cidrIpRange` – IPv4 address range in CIDR format. E.g., `10.0.0.0/24.`
@param – `ipAddress` – The IPv4 IP address to test for inclusion in `cidrIpRange.`

`C.Net.isIpV4()`
(method) `Net.isIpV4(value: string): boolean`
Determine if the value supplied is an IPv4 address. (Available in Cribl Stream v.3.1.2+.)
@param – `value` – the IP address to test.

`C.Net.isIpV6()`
(method) `Net.isIpV6(value: string): boolean`
Determine if the value supplied is an IPv6 address. (Available in Cribl Stream v.3.1.2+.)
@param – `value` – the IP address to test.

`C.Net.isIpV4AllInterfaces()`

(method) `Net.isIpV4AllInterfaces(value: string): boolean`

Determine if the value supplied is the IPv4 all-interfaces address, typically used to bind to all IPv4 interfaces – i.e., '0.0.0.0'. (Available in Cribl Stream v.3.1.2+.) @param – `value` – the IP address to test.

`C.Net.isIpV6AllInterfaces()`

(method) `Net.isIpV6AllInterfaces(value: string): boolean`

Determine if the value supplied is an IPv6 all-interfaces address, typically used to bind to all IPv6 interfaces – one of: '::', '0:0:0:0:0:0:0:0', or '0000:0000:0000:0000:0000:0000:0000:0000'. (Available in Cribl Stream v.3.1.2+.) @param – `value` – the IP address to test.

`C.Net.ipv6Normalize()`

(method) `Net.ipv6Normalize(address: string): string`

Normalize an IPV6 address based on RFC draft-ietf-6man-text-addr-representation-04.
@param – `address` – the IPV6 address to normalize.

`C.Net.isPrivate()`

(method) `Net.isPrivate(address: string): string`

Determine whether the supplied IPv4 address is in the range of private addresses per RFC1918.
@param – `address` – address to test.

C.os – System Functions

`C.confVersion`

Returns Cribl Stream config version.

`C.os.hostname()`

Returns hostname of the system running this Cribl Stream instance.

C.Schema – Schema Functions

`C.Schema()`

(property) `Schema: (id: string) => SchemaValidator`

(method) `SchemaValidator.validate(data: any): boolean`

Validates the given object against the schema.
@param – `data` – object to be validated.
@returns – `true` when schema is valid; otherwise, `false`.
Example: `C.Schema('schema1').validate(myField)` will validate if `myField` object conforms to `schema1`.

See Schema Library for more details.

C.Secret – Secrets-Management Functions

`C.Secret()`
(method) `Secret: (id: string, type?: string): ISecret`
(method) `Secret(id: string, type: 'keypair') => IPairSecret`
(method) `Secret(id: string, type: 'text') => ITextSecret`
(method) `Secret(id: string, type: 'credentials') => ICredentialsSecret`
Returns a secret matching the specified ID.
@param `id` – ID of the secret.
@param `type` – optional type of the secret.

Examples:

- `C.Secret('victorias', 'text')` will return a text secret with ID `victorias` (or with undefined, if no such secret exists).
- `C.Secret('api_key', 'keypair').secretKey`
- `C.Secret('secret_hash', 'text').value`
- `C.Secret('user_pass', 'credentials').password`

Common returned attributes for `ISecret` objects:

- `secretType` – one of `keypair`, `text`, or `credentials`.
- `description` (optional) – the secret description.
- `tags` (optional) – a comma separated list of tags.

Additional returned attributes for `IPairSecret` objects:

- `apiKey` – the API key value
- `secretKey` – the Secret key value

Additional returned attributes for `ITextSecret` objects:

- `value` – the text value

Additional returned attributes for `ICredentialsSecret` objects:

- `username` – the username value
- `password` – the password value

See Securing Cribl Stream > Secrets for more details.

C.Text – Text Functions

`C.Text.entropy()`

(method) `Text.entropy(bytes: any): number`

Computes the Shannon entropy of the given buffer or string.

@param – `bytes` – value to undergo Shannon entropy computation.

@returns – the entropy value; or `-1` in case of an error.

`C.Text.hashCode()`

(method) `Text.hashCode(val: string | Buffer | number): number`

Computes hashcode (djb2) of the given value.

@param – `val` - value to be hashed.

@returns – hashcode value.

`C.Text.isASCII()`

(method) `Text.isASCII(bytes: any): boolean`

Checks whether all bytes or chars are in the ASCII printable range.

@param – `bytes` – value to check for character range.

@returns – `true` if all chars/bytes are within ASCII printable range; otherwise, `false`.

`C.Text.isUTF8()`

(method) `Text.isUTF8(bytes: any): boolean`

Checks whether the given Buffer contains valid UTF8.

@param – `bytes` – bytes to check.

@returns – `true` if bytes are UTF8; otherwise, `false`.

`C.Text.parseWinEvent`

(method) `C.Text.parseWinEvent(xml: string, nonValues?: string[]): any`

Parses an XML string representing a Windows event into a compact, prettified JSON object. Works like `C.Text.parseXml` , but with Windows events, produces more-compact output. For a usage example, see Reducing Windows XML Events.

@param – `xml` – an XML string; or an event field containing the XML.

@param – `nonValues` – array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `['-']`, meaning that elements whose value equals `-` will be discarded.

@returns – an object representing the parsed Windows Event; or `undefined` if the input could not be parsed.

C.Text.parseXml

(method) `C.Text.parseXml(xml:string, keepAttr?:boolean, keepMetadata?:boolean, nonValues?:string[]): any`

Parses an XML string and returns a JSON object. Can be used with Eval Function to parse XML fields contained in an event, or with ad hoc XML.

@param – `xml` – XML string, or an event field containing the XML.

@param – `keepAttr` – whether or not to include attributes in the returned object. Defaults to `true`.

@param – `keepMetadata` – whether or not to include metadata found in the XML. The `keepAttr` parameter must be set to `true` for this to work. Defaults to `false`. (Eligible metadata includes namespace definitions and prefixes, and XML declaration attributes such as encoding, version, etc.)

@param – `nonValues` – array of string values. Elements whose value equals any of the values in this array will be omitted from the returned object. Defaults to `[]` (empty array), meaning discard no elements.

@returns – an object representing the parsed XML; or `undefined` if the input could not be parsed. An input collection of elements will be parsed into an array of objects.

C.Text.relativeEntropy()

(method) `Text.relativeEntropy(bytes: any, modelName?: string): number`

Computes the relative entropy of the given buffer or string.

@param – `bytes` – Value whose relative entropy to compute.

@param – `modelName` – Optionally, override the default `$CRIBL_HOME/data/lookups/model_relative_entropy_top_domains.csv` model used to test the input. Create a custom lookup file with the same column and value structure as the default, and store it in the same path, as `model_relative_entropy_<custom-name>.csv`. To reference it, pass your `<custom-name>` substring as the `modelName` parameter.

@returns – The relative entropy value, or `-1` in case of an error.

> When using `modelName` in a distributed deployment, the corresponding paths are `$CRIBL_HOME/groups/<worker–group-name>/data/lookups/`. Creating your custom lookup file via Cribl Stream's UI will automatically set the appropriate paths.

C.Time – Time Functions

C.Time.adjustTZ()

(method) `Time.adjustTZ(epochTime: number, tzTo: string, tzFrom?: string): number`

Adjust a timestamp from one timezone to another.

@param – `epochTime` – UNIX epoch time.

@param – `tzTo` – timezone to adjust to.

@param – `tzFrom` – optional timezone of the timestamp.

@returns – the adjusted timestamp, in UNIX epoch time (ms).

> Cribl relies on this timezone-support package, and on this list of TZ Database Time Zones.

`C.Time.clamp()`

(method) `Time.clamp(date, earliest, latest, defaultDate?): number`

Constrains an event's parsed timestamp to realistic earliest and latest boundaries.

@param – `date` – Timestamp originally parsed from event, in UNIX epoch time (ms) or JavaScript Date format.

@param – `earliest` – earliest allowable timestamp, in UNIX epoch time (ms) or JS Date format.

@param – `latest` – latest allowable timestamp, in UNIX epoch time (ms) or JS Date format.

@param – `defaultDate` – optional default date, in UNIX epoch time (ms) or JS Date format, to substitute for values outside the `earliest` or `latest` boundaries.

`C.Time.strftime()`

(method) `Time.strftime(date: number | Date, format: string, utc?: boolean): string`

Format a Date object or number as a time string, using strftime specifier.

@param – `date` – Date object or number (seconds since epoch) to format.

@param – `format` – specifier to use to format the date.

@param – `utc` – whether to output the time in UTC, rather than in local timezone.

@returns – representation of the given date.

`C.Time.strptime()`

(method) `Time.strptime(str: string, format: string, utc?: boolean, strict?: boolean): Date`

Extract time from a string using strptime specifier.

@param – `str` – string to parse to a timestamp (see strict flag).

@param - `format` – `strptime` specifier.

@param – `utc` – whether to interpret times as UTC, rather than as local time.

@param – `strict` – whether to return `null` if there are any extra characters after timestamp.

@returns – a parsed Date object, if successful; otherwise, `null` if the specifier did not match.

`C.Time.timestampFinder()`

(method) `Time.timestampFinder(utc?: boolean).find(<source-field>): AutoTimeParser`

Extract time from the specified `<source-field>`, using the same algorithm as the Auto Timestamp Function and the Event Breaker Function.

@param – `utc` – whether to output the time in UTC, rather than in local timezone.

@param – `<source-field>` – the field in which to search for the time.

@returns – representation of the extracted time.

C.vars – Global Variables

See Global Variables Library for more details.

C.version – Cribl Stream Version

(property) `version: string`
Cribl Stream Version.

;

10.4. 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
```

Note that `$CRIBL_VOLUME_DIR`, when set, overrides `$CRIBL_HOME`.

Commands Available

To see a list of available commands, enter `./cribl` alone (or the equivalent `./cribl help`). To execute a command, or to see its required parameters, enter `./cribl <command>`.

`help`

Displays a list of commands with a description (help) for each. Defaults to a selection of generally useful commands.

Usage

```
./cribl help [-a]
```

Options

```
-a               - Display the list of all commands, except for `scope`.
```

Sample Response

```
Software version: 3.4.0
Usage: [sub-command] [options] [args]

Commands:
help              - Display help
mode-edge         - Configure instance in Edge mode
mode-managed-edge - Configure instance in Managed-Edge mode
mode-master       - Configure instance in Leader mode
mode-single       - Configure instance in Single-Instance mode
mode-worker       - Configure instance in Worker mode
reload            - Reload instance
restart           - Restart instance
start             - Start instance
status            - Display status
stop              - Stop instance
version           - Print version

auth              - Authentication
boot-start        - Enable/Disable boot-start
diag              - Manage diagnostics bundles
git               - Manage Worker Groups config
keys              - Manage encryption keys
nc                - Listen on a port for traffic and output stats and data
node              - Execute a JavaScript file
pack              - Manage Cribl Packs
pipe              - Feed stdin to a pipeline
vars              - Manage global variables
```

> As of version 3.0, Cribl Stream's former "master" application components are renamed "leader." As long as some legacy terminology remains within CLI commands/options, configuration keys/values, and environment variables, this document will reflect that.

`mode-master`

Configures Cribl 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.
[-k <privKeyPath>]      – Server path containing the private key (in PEM format) to
use. Can reference $ENV_VARS.
[-c <certPath>]         – Server path containing certificates (in PEM format) to
use. Can reference $ENV_VARS.
[-u <authToken>]        - Optional authentication token to include as part of the
connection header.
[-i <ipWhitelistRegex>] – Regex matching IP addresses that are allowed to establish
a connection.
[-r <resiliency>]       – Resiliency mode for the Leader. Accepts agruments: none,
failover.
[-v <failoverVolume>]   – If the -r option is set to failover, this defines the
volume to use as shared storage.
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

mode-single

Configures Cribl Stream as a single-instance deployment.

Usage

```
./cribl mode-single [--help]
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

mode-edge

Configures Cribl Edge as a single-instance deployment.

Usage

```
./cribl mode-edge [--help]
```

Sample Response

```
Settings updated.
```

`mode-worker`

Configures Cribl Stream as a Worker instance.

Usage

`./cribl mode-worker -H <host> -p <port> <options> <args>`

The `-H <host> -p <port>` parameters are required.

Options

```
-H <host>          – Leader Node's Hostname or IP address.
-p <port>          – Leader Node's cluster communications port (defaults to 4200).
-S                 – Set secure communication with the Leader using TLS.
UI-generated update scripts automatically include this option.
[-n <certName>]    – Name of saved certificate.
[-k <privKeyPath>] – Server path containing the private key (in PEM format) to use.
Can reference $ENV_VARS.
[-c <certPath>]    – Server path containing certificates (in PEM format) to use. Can
reference $ENV_VARS.
[-u <authToken>]   – Authentication token to include as part of the connection
header. By default, this token is included and is set to 'criblmaster'.
[-e <envRegex>]    – Regex that selects environment variables to report to Leader.
[-t <tags>]        – Tag values to report to Leader.
[-g <group>]       – Worker Group/Fleet to report to Leader.
```

Sample Response

```
Settings updated.
You will need to restart Cribl Stream before your changes take full effect.
```

`mode-managed-edge`

Configures Cribl Edge as an Edge Node.

Usage

```
./cribl mode-managed-edge -H <host> -p <port> <options> <args>
```

The `-H <host> -p <port>` parameters are required.

Options

```
-H <host>          – Leader Node's Hostname or IP address.
-p <port>          – Leader Node's cluster communications port (defaults to 4200).
-S <tls>           – Enables secure TLS communication between Leader and Worker/Edge
Nodes. Accepts: ["true", "false"]. UI-generated version-update scripts automatically
set this option to true.
[-n <certName>]    – Name of saved certificate.
[-k <privKeyPath>] – Server path containing the private key (in PEM format) to use.
Can reference $ENV_VARS.
[-c <certPath>]    – Server path containing certificates (in PEM format) to use. Can
reference $ENV_VARS.
[-u <authToken>]   – Authentication token to include as part of the connection
header. By default, this token is included and is set to 'criblmaster'.
[-e <envRegex>]    – Regex that selects environment variables to report to Leader.
[-t <tags>]        – Tag values to report to Leader.
[-g <group>]       – Worker Group/Fleet to report to Leader.
```

Sample Response

```
Settings updated.
```

pack

Manages Cribl Packs.

Usage

```
./cribl pack <sub-command> <options> <args>
```

Sub-commands and Options

```
export              - Export Cribl Packs, args:
   -m <mode>        - Mode to export. Accepts: merge_safe, merge, default_only.
  [-o <filename>]   - Where to export the pack on disk.
  [-n <name>]       - Name to override the installed pack's name on export.
  [-g <group>]      - The Worker Group/Fleet to execute within
install             - Install a Cribl Pack, args:
  [-d ]             - Run install in debug.
  [-f ]             - Force install.
  [-n <name>]       - Name of the pack to install; defaults to source.
  [-g <group>]      - The Worker Group/Fleet to execute within.
list                - List Cribl Packs, args:
  [-v ]             - Display all pack info.
  [-g <group>]      - The Worker Group/Fleet to execute within.
uninstall           - Uninstall a Cribl Pack, args:
  [-d ]             - Run uninstall in debug.
  [-g <group>]      - The Worker Group/Fleet to execute within.
upgrade             - Upgrade a Cribl Pack, args:
  [-d ]             - Run upgrade in debug.
  [-s <source>]     - Provide the pack source.
  [-m <minor>]      - Only upgrade to minor version.
  [-g <group>]      - The Worker Group/Fleet to execute within.
```

Sample Response

```
id          version  spec  displayName    author       description
source
------------------------------------------------------------------------------------
------------------------------------------
HelloPacks  1.0.0    ----  Hello, Packs!  Cribl, Inc.  A sample pack with a simple
example  file:/opt/cribl/default/HelloPacks
```

`reload`

Reloads Cribl 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 <oldHost>]    - undefined
  [-H <host>]       - Host URL (e.g. http://localhost:9000)
  [-u <username>]   - Username
  [-p <password>]   - Password
  [-f <file>]       - File with credentials
logout              - Log out from Cribl Stream/Edge
```

Login Examples

Launch interactive login:

```
$CRIBL_HOME/bin/cribl auth login
```

Append credentials as command arguments:

```
$CRIBL_HOME/bin/cribl auth login -h <url> -u <username> -p <password>
```

> All `-h` and `host` arguments are optional, provided that the API host and port are listed in the `cribl.yml` file's `api:` section.

Provide credentials in environment variables:

```
CRIBL_HOST=<url> CRIBL_USERNAME=<username> CRIBL_PASSWORD=<password>
$CRIBL_HOME/bin/cribl auth login
```

Provide credentials in a file:

```
$CRIBL_HOME/bin/cribl auth login -f <path/to/file>
```

--

Corresponding file contents:

```
host=<url>
username=<username>
password=<password>
```

`boot-start`

Enables or disables Cribl 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
list                 - List existing Cribl Stream/Edge diagnostic bundles

send                 - Send Cribl Stream/Edge diagnostics bundle to Cribl Support,
args:
   -c <caseNumber>   - Cribl Support Case Number
  [-p <path>]        - Diagnostic bundle path (if empty then new bundle will be
created)
```

Sample Response

```
Created @[product] diagnostic bundle at /opt/cribl/diag/logstream-zedborcdb72f-
20210820T204405.tar.gz
```

`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:
  [-c <keyclass>] - key class to set for the key
  [-k <kms>]      - KMS to use, must be configured, see cribl.yml
  [-e <expires>]  - expiration time, epoch time
  [-i ]           - use an initialization vector
   -g <group>     - Group ID
list               - List encryption keys, args:
   -g <group>     - Group ID
```

Sample Response

```
Adding key succeeded. Key count=1
```

nc

Listens on a port for traffic, and outputs stats and data. (Netcat-like utility.)

Usage

```
./cribl nc -p <port> <options> <args>
```

Options

```
 -p <port>            - Port to listen on
[-s <statsInterval>] - Stats output interval (ms), use 0 to disable
[-u]                 - Listen on UDP port instead
[-o]                 - Output received data to stdout
[-t <throttle>]      - throttle rate in (unit)/sec, where units can be KB,MB,GB and
TB
```

Sample Response

```
2021-08-20T22:44:30.457Z - starting server on 0.0.0.0:9999
2021-08-20T22:44:30.462Z - server listening 0.0.0.0:9999
2021-08-20T22:44:31.461Z - messages: 0, socks: 0, thruput: 0MBps
2021-08-20T22:44:32.466Z - messages: 0, socks: 0, thruput: 0MBps
...
2021-08-20T22:44:39.212Z - got connection: 127.0.0.1:37190
2021-08-20T22:44:39.213Z - got connection: 127.0.0.1:37192
```

node

Run with no options, displays a command prompt, as shown here:

```
>
```

To execute a JavaScript file, you can enter path/filename at the prompt.

With the `-v` option, prints the version of NodeJS that is running.

With `-e` , evaluates a string. Write to console to see the output, for example:

```
./cribl node -e 'console.log(Date.now())'
1629740667695
```

Usage

`./cribl node <options> <args>`

Options

```
[-e <eval>] - String to eval
[-v]        - Prints NodeJS version
```

Sample Response

```
v14.15.1
```

`pipe`

Feeds stdin to a pipeline.

Usage

`./cribl pipe -p <pipelineName> <options> <args>`

Examples:

```
cat sample.log |  ./cribl pipe -p <pipelineName>
cat sample.log |  ./cribl pipe -p <pipelineName> 2>/dev/null
```

Options

```
 -p <pipeline>    - Pipeline to feed data thru
[-d]              - Include dropped events
[-c <cpuProfile>] - Perform CPU profiling
[-a <pack>]       - Optional Cribl Pack context
```

Sample Response

```
...
{"time":"2021-08-
20T20:37:00.017Z","cid":"api","channel":"commands","level":"info","message":"creating
new pipeline","id":"main","conf":{"asyncFuncTimeout":1000,"functions":
[{"id":"eval","disabled":false,"filter":"true","conf":{"add":
[{"name":"cribl","value":"'yes'"}],"remove":[]}}]}}
{"time":"2021-08-
20T20:37:00.019Z","cid":"api","channel":"pipe:main","level":"info","message":"start
loading and initializing functions","count":1}
{"time":"2021-08-
20T20:37:00.021Z","cid":"api","channel":"pipe:main","level":"info","message":"finished
loading and initializing functions","count":1}
{"time":"2021-08-
20T20:37:00.022Z","cid":"api","channel":"commands","level":"info","message":"START
pushing stdin events","id":"main"}
{"time":"2021-08-
20T20:37:00.028Z","cid":"api","channel":"GrokMgr","level":"info","message":"loaded
grok patterns","count":152}
...
```

`scope`

Greps your apps by the syscalls. Executes immediately.

See the AppScope CLI Reference for usage and examples.

`vars`

Manages Cribl 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"
  }
]
```

;

10.5. 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:4200` . See Formatting Notes below.
`CRIBL_DIST_MODE`	`worker` or `master` . Defaults to `worker` **iff** `CRIBL_DIST_MASTER_URL` is present.
`CRIBL_HOME`	Auto setup on startup. Defaults to parent of `bin` directory.
`CRIBL_CONF_DIR`	Auto setup on startup. Defaults to parent of `bin` directory.
`CRIBL_NOAUTH`	Disables authentication. Careful here!!
`CRIBL_TMP_DIR`	Defines the root of a temporary directory. See Formatting Notes below.
`CRIBL_VOLUME_DIR`	Sets a directory that persists modified data between different containers or ephemeral instances.
`CRIBL_DIST_WORKER_PROXY`	Communicate to the Leader Node via a SOCKS proxy.See Formatting Notes below.
`CRIBL_BOOTSTRAP`	Quickstart a Cribl instance by configuring this variable.
`CRIBL_BOOTSTRAP_HOST`	Host name for connecting to the Leader Node when setting up a new 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> $CRIBL_HOME/bin/cribl auth login`

Formatting 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`.

`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`.

Adding a Second Leader Node

You can configure a second Leader Node via the following environment variables.

NAME	PURPOSE
`CRIBL_DIST_MASTER_RESILIENCY=failover`	Sets the Leader's `Resiliency` to `Failover` mode.
`CRIBL_DIST_MASTER_FAILOVER_VOLUME=/tmp/shared`	Sets the location of the NFS directory to support Leader failover.
`CRIBL_DIST_MASTER_FAILOVER_MISSED_HB_LIMIT`	Determines how many Lease refresh periods elapse before the standby Nodes attempt to promote themselves to primary. Cribl recommends setting this to `3`.
`CRIBL_DIST_MASTER_FAILOVER_PERIOD`	Determines how often the primary Leader refreshes its hold on the Lease file. Cribl recommends setting this to `5s`.
`CRIBL_INSTANCE_HOME`	In `Failover` mode, this variable points to the Leader Node's `root` directory, as opposed to the shared volume. It is used to access `$CRIBL_INSTANCE_HOME/local/_system/instance.yml`. Outside of `Failover` mode, it's the same value as `CRIBL_CONF_DIR`.

GitOps

Cribl 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`	Controls which GitOps workflow to use – one of: `none` , `push` , or `pull` .
`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_WORKER_ID`	Passed to Worker processes.
`CRIBL_GROUP_ID`	Passed to ConfigHelper processes to identify Worker Groups.
`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_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_FS_ROOT`	Location of the host OS' filesystem when mounting in a container. Defaults to `/hostfs` .

NAME	PURPOSE
`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`.

;

10.6. 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`
Pipelines Configuration	`(default|local)/cribl/pipelines/<pname>` Each Pipeline's conf is contained therein.
Routes Configuration	`(default|local)/cribl/pipelines/routes.yml`
Functions	`(default|local)/cribl/functions/<function_name>` Each function's code, conf is contained therein.
Functions Configuration	`(default|local)/cribl/functions/<function_name>/...` Each function's conf is contained therein.
Roles Configuration	`(default|local)/cribl/roles.yml` RBAC Role definitions. See roles.yml.
Policies Configuration	`(default|local)/cribl/policies.yml` RBAC Policy definitions. See policies.yml.

Configurations and Restart

- Configuration changes resulting from most UI interactions – for instance, changing the order of Functions in a Pipeline, or changing the order of Routes – **do not require restarts**.
- Some configuration changes in the **Settings** UI **do require restarts**. You will be prompted to confirm before restarting.
- All direct edits to configuration files in `(bin|local|default)/cribl/...` **will require restarts**.
- Worker Nodes/Edge Nodes might temporarily disappear from the Leader's **Workers** tab while restarting.
- When using the Cribl App for Splunk, changes to Splunk configuration files might or might not require restarts. Please check current Splunk docs.

Configuration Layering and Precedence

Similar to most *nix systems, Cribl configurations in `local` take precedence over those in `default`. There is no layering of configuration files.

Editing Configuration Files Manually

When config files **must** be edited manually, save all changes in `local`.

;

10.6.1. cribl.yml

`cribl.yml` contains settings for configuring API and other system properties.

$CRIBL_HOME/default/cribl/cribl.yml

```
auth: # [object] Authentication Settings
  type: # [string] Type - Select one of the supported authentication providers.

  # -------------- if type is ldap ---------------

  secure: # [boolean] Secure - Enable to use a secure ldap connection (ldaps://),
disable for unsecure (ldap://) connection.
  ldapServers: # [array of strings] LDAP servers - List of LDAP servers, each entry
should contain host:port (e.g., localhost:389)
  bindDN: # [string] Bind DN - Distinguished name of entity to authenticate with
LDAP server, e.g., 'cn=admin,dc=example,dc=org'
  bindCredentials: # [string] Password - Distinguished Name password used to
authenticate with LDAP server
  searchBase: # [string] User search base - Starting point to search LDAP for users,
e.g., 'dc=example,dc=org'
  usernameField: # [string] Username field - LDAP user search field, e.g. cn or uid
  searchFilter: # [string] User search filter - LDAP search filter to apply when
finding user, e.g. (&(group=admin)(!(department=123*)))
  groupSearchBase: # [string] Group search base - Starting point to search LDAP for
groups, e.g., 'dc=example,dc=org'
  groupMemberField: # [string] Group member field - LDAP group search field, e.g.
member
  groupSearchFilter: # [string] Group search filter - LDAP search filter to apply
when finding group, e.g. (&(cn=cribl*)(objectclass=group))
  groupField: # [string] Group name field - LDAP group field, e.g. cn
  connectTimeout: # [number] Connection timeout (ms)
  rejectUnauthorized: # [boolean] Reject unauthorized - Valid for secure LDAP
connections, set true to reject unauthorized server certificates.
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if
LDAP authentication is down or mis-configured. Defaults to false.
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    mapping: # [object] Mapping - Group(s)-to-role(s) mappings

  # --------------------------------------------------------


  # -------------- if type is saas ---------------

  issuer: # [string] Issuer - Issuer from which to accept and validate JWT tokens.
  tenantId: # [string] Organization ID - The organization ID within which this
instance is running.
  loginUrl: # [string] Login URL - The URL to redirect unauthenticated users to.

  # --------------------------------------------------------


  # -------------- if type is splunk ---------------

  host: # [string] Host - Hostname or address of Splunk instance that provides
authentication. Defaults to localhost.
  port: # [number] Port - Port of Splunk instance that provides authentication.
Defaults to 8089.
  ssl: # [boolean] SSL - Whether SSL is enabled on Splunk instance that provides
authentication. Defaults to Yes.
  fallback: # [boolean] Fallback on fatal error - Attempt local authentication if
Splunk is unsuccessful. Defaults to false.
  groups: # [object]
```

```
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    mapping: # [object] Mapping - Group(s)-to-role(s) mappings

  # ----------------------------------------------------------


  # -------------- if type is openid ---------------

  name: # [string] Provider name - The name of the identity provider service. Manual
entries are also allowed.
  audience: # [string] Audience - The Audience from provider configuration. This
will be the base URL of your Cribl server, i.e.: https://yourDomain.com:9000
  client_id: # [string] Client ID - The client_id from provider configuration.
  client_secret: # [string] Client secret - The client_secret provider
configuration.
  scope: # [string] Scope - Space-separated list of authentication scopes. Default:
openid profile email.
  auth_url: # [string] Authentication URL - The full path to the provider's
authentication endpoint. Be sure to configure the callback URL at provider as
<yourDomainUrl>/api/v1/auth/authorization-code/callback, e.g.
https://yourDomain.com:9000/api/v1/auth/authorization-code/callback
  token_url: # [string] Token URL - The full path to the provider's access token
URL.
  userinfo_url: # [string] User Info URL - The full path to the provider's user info
url. If not provided, Stream will attempt to gather user info from the ID token
returned from the Token URL.
  logout_url: # [string] Logout URL - The full path to the provider's logout URL.
Leave blank if the provider does not support logout or token revocation.
  userIdExpr: # [string] User identifier - Expression used to derive userId from the
id_token returned by the openId provider.
  rejectUnauthorized: # [boolean] Validate certs - Validate certificates; set to
false to allow insecure self-signed certificates.
  filter_type: # [string] Filter type - Optional method for limiting access per
user.
  groupField: # [string] Group name field - Field on the id_token that contains the
user groups.
  fallback: # [boolean] Allow local auth - Allows locally configured users to log in
to Stream.
  groups: # [object]
    default: # [string] Default role - Default role assigned to groups not
explicitly mapped to a role.
    mapping: # [object] Mapping - Group(s)-to-role(s) mappings

  # ----------------------------------------------------------

system: # [object]
  upgrade: # [string]
  restart: # [string]
  installType: # [string]
workers: # [object]
  count: # [number]
  memory: # [number]
tls: # [object] Default TLS Settings
  minVersion: # [string] Minimum TLS version - Minimum TLS version. Defaults to
TLSv1.
  maxVersion: # [string] Maximum TLS version - Maximum TLS version. Defaults to
TLSv1.3.
  defaultCipherList: # [string] Default cipher list - Default suite of enabled and
disabled TLS ciphers. Defaults to:
      ECDHE-RSA-AES128-GCM-SHA256:
      ECDHE-ECDSA-AES128-GCM-SHA256:
      ECDHE-RSA-AES256-GCM-SHA384:
      ECDHE-ECDSA-AES256-GCM-SHA384:
      DHE-RSA-AES128-GCM-SHA256:
      ECDHE-RSA-AES128-SHA256:
      DHE-RSA-AES128-SHA256:
      ECDHE-RSA-AES256-SHA384:
      DHE-RSA-AES256-SHA384:
      ECDHE-RSA-AES256-SHA256:
      DHE-RSA-AES256-SHA256:
      HIGH:
      !aNULL:
      !eNULL:
      !EXPORT:
      !DES:
      !RC4:
      !MD5:
      !PSK:
      !SRP:
      !CAMELLIA
  defaultEcdhCurve: # [string] ECDH curve - The curve name, or a colon-separated
list of curve NIDs or names, to use for ECDH key agreement. For example:
'Pâ¯💰521:Pâ¯💰384:Pâ¯💰256'. Defaults to 'auto'.
  rejectUnauthorized: # [boolean] Validate server certs - Whether to validate server
certificates globally. Set to false to allow self-signed certs.
proxy: # [object]
  useEnvVars: # [boolean]
git: # [object]
  branch: # [string] Branch - The branch to track in your Stream deployment's git
repository.
  gitOps: # [string] GitOps workflow - The GitOps workflow for managing Stream's
config.
  commitDeploySingleAction: # [boolean] Collapse actions - When enabled, Commit &
Deploy will be collapsed into a single action. If you've configured a remote, Commit
& Git Push will also be collapsed. Your DefaultÂ CommitÂ Message below will be used
for all commits.
  defaultCommitMessage: # [string] Default commit message - Enter a default message
to use for all commits.
  remote: # [string] Remote URL - Enter remote git repo's URL.
  authType: # [string] Authentication type - Git authentication type.

  # -------------- if authType is ssh ---------------

  sshKey: # [string] SSH private key - Enter SSH private key (without passphrase) to
use for authentication on remote git repo.
  strictHostKeyChecking: # [boolean] SSH strict host key checking - Validate key
against known hosts, to prevent spoofing or impersonation attacks. For details, see
"VerifyingÂ Host Keys" [here](https://linux.die.net/man/1/ssh).

  # -----------------------------------------------------



  # -------------- if authType is basic ---------------

  user: # [string] User - Username for authentication.
  password: # [string] Password - Password for authentication. (With GitHub, use a
personal access token.)

  # -----------------------------------------------------
  autoAction: # [string] Scheduled global actions - Global git actions to run automatically on a schedule.

  # -------------- if autoAction is commit ---------------

  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action on.
  autoActionMessage: # [string] Commit message - Default scheduled commit message.

  # --------------------------------------------------------


  # -------------- if autoAction is push ---------------

  autoActionSchedule: # [string] Schedule - Cron schedule to run selected git action on.

  # --------------------------------------------------------

  timeout: # [number] Git timeout - Max time (milliseconds) to wait for git processes before killing them, 0 to wait indefinitely
```

Example `cribl.yml`:

$CRIBL_HOME/default/cribl/cribl.yml

```yaml
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
```

;

10.6.2. 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 data to apply rule to. To test your sample, use the maximize icon on the right.
      type: # [string] Event Breaker Type - Event Breaker Type
      timestampAnchorRegex: # [string] Timestamp Anchor - Regex to match before attempting timestamp extraction. Use $ (end of string anchor) to not perform extraction.
      timestamp: # [object] Timestamp Format - Auto, manual format (strptime) or current time.
      type: # [string] Timestamp Type
      length: # [number] Length
      format: # [string] Format
      timestampTimezone: # [string] Default timezone - Timezone to assign to timestamps without timezone info.
      timestampEarliest: # [string] Earliest timestamp allowed - The earliest timestamp value allowed relative to now. E.g., -42years. Parsed values prior to this date will be set to current time.
      timestampLatest: # [string] Future timestamp allowed - The latest timestamp value allowed relative to now. E.g., +42days. Parsed values after this date will be set to current time.
      maxEventBytes: # [number] Max Event Bytes - The maximum number of bytes that an event can be before being flushed to the pipelines
      fields: # [array] Fields - Key value pairs to be added to each event.
      - name: # [string] Name - Field Name.
        value: # [string] Value Expression - JavaScript expression to compute fields value (can be constant).
      disabled: # [boolean] Disabled - Allows breaker rule to be enabled or disabled, default is enabled.
```

;

10.6.3. 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. Optional.
  cert: # [string] Certificate - Drag/drop or upload host certificate, in PEM/Base64 format. Or paste its contents here.
  privKey: # [string] Private key - Certificate private key.
  passphrase: # [string] Passphrase - Passphrase. Optional.
  ca: # [string] CA certificate - Optionally, drag/drop or upload all CA certificate(s) in PEM/Base64 format. Or paste certs' contents here. Certs can be used for client and/or server auth.
  inUse: # [array of strings] Referenced - List of configurations referencing this certificate.
```

;

10.6.4. 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 for this group
  onPrem: # [boolean] On prem - Whether the group accepts on-prem or cloud worker nodes (this field is only available on cloud deployments)
  isFleet: # [boolean] Is Fleet - Fleet groups only manage edge nodes.
  workerRemoteAccess: # [boolean] UI access - Enable authenticated viewing of Workers' UI from the Leader.
```

;

10.6.5. 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
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # -------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttle
while writing to an output. Also takes values with multiple-byte units, such as KB, MB,
GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.
    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  kafka_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Kafka brokers to use, eg.
localhost:9092
    topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: To
optimize performance, Cribl suggests subscribing each Kafka Source to only a single
topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance belongs t
default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from earliest
available data, relevant only during initial subscription.
    kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
      disabled: # [boolean] Disabled - Enable Schema Registry

      # -------------- if disabled is false ---------------

      schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Conflue
Schema Registry, i.e: http://localhost:8081
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the
system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in which t
find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ----------------------------------------------------------


      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for a
successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when connecting
brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable Authentication

      # -------------- if disabled is false ---------------
      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

      # --------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the
system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in which
find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which to
find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # --------------------------------------------------------

    sessionTimeout: # [number] Session timeout (ms) -
      Timeout used to detect client failures when using Kafka's group management
facilities.
      If the client sends the broker no heartbeats before this timeout expires,
      the broker will remove this client from the group, and will initiate a rebalance.
      Value must be between the broker's configured group.min.session.timeout.ms and
group.max.session.timeout.ms.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_session.timeout.ms).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time for each worker to join the group after a rebalance has begu
      If the timeout is exceeded, the coordinator broker will remove the worker from th
group.
      See details [here]
(https://kafka.apache.org/documentation/#connectconfigs_rebalance.timeout.ms).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time between heartbeats to the consumer coordinator when using Kafka's
group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3 of t
sessionTimeout value.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_heartbeat.interval.ms).
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
```

```
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  http_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by any
client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
```

```
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    criblAPI: # [string] Cribl HTTP Event API - Absolute path on which to listen for th
Cribl HTTP API requests. At the moment, only _bulk (default /cribl/_bulk) is available.
Use empty string to disable.
    elasticAPI: # [string] Elasticsearch API endpoint (Bulk API) - Absolute path on whi
to listen for the Elasticsearch API requests. At the moment only _bulk (default
/elastic/_bulk) is available. Use empty string to disable
    splunkHecAPI: # [string] Splunk HEC Endpoint - Absolute path on which listen for th
Splunk HTTP Event Collector API requests. Use empty string to disable.
    splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC
acknowledgements
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
```

```
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates tc
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ---------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    authTokens: # [array] Auth tokens - Shared secrets to be provided by any Splunk
forwarder. IfÂ empty, unauthed access is permitted.
      - token: # [string] Token - Shared secrets to be provided by any Splunk forwarder
IfÂ empty, unauthed access is permitted.
        description: # [string] Description - Optional token description
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_search_input: # [object]
    searchHead: # [string] Search head - Search head base URL, can be expression, defau
is https://localhost:8089.
    search: # [string] [required] Search - Enter Splunk search here. For example:
'index=myAppLogs level=error channel=myApp' OR '| mstats avg(myStat) as myStat WHERE
index=myStatsIndex.'
    earliest: # [string] Earliest - The earliest time boundary for the search. Can be a
exact or relative time. For example: '2022-01-14T12:00:00Z' or '-16m@m'
    latest: # [string] Latest - The latest time boundary for the search. Can be an exac
or relative time. For example: '2022-01-14T12:00:00Z' or '-1m@m'
    cronSchedule: # [string] [required] Cron schedule - A cron schedule on which to run
```

```
this job.
    endpoint: # [string] [required] Search endpoint - REST API used to create a search.
    outputMode: # [string] [required] Output mode - Format of the returned output
    endpointParams: # [array] Endpoint parameters - Optional request parameters to send
to the endpoint.
      - name: # [string] Name - Parameter name
        value: # [string] Value - JavaScript expression to compute the parameter's valu
normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quote
(e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as
strings.
    endpointHeaders: # [array] Endpoint headers - Optional request headers to send to t
endpoint.
      - name: # [string] Name - Header Name
        value: # [string] Value - JavaScript expression to compute the header's value,
normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quote
(e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as
strings.
    logLevel: # [string] Log level - Collector runtime log Level (verbosity).
    requestTimeout: # [number] Request timeout (seconds) - HTTP request inactivity
timeout, use 0 to disable
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS looku
When a DNS server returns multiple addresses, this will cause Stream to cycle through
them in the order returned.
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should chec
in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Alive
Time periods before an inactive worker will have its job subscription revoked.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    authType: # [string] Authentication type - Splunk Search authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # --------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # --------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # --------------------------------------------------------
    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text secre

    # -----------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # -----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # -----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_hec_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array] Auth tokens - Shared secrets to be provided by any client
(Authorization: <token>). If empty, unauthed access is permitted
      - token: # [string] Token - Shared secret to be provided by any client
(Authorization: <token>).
```

```
        description: # [string] Description - Optional token description
        metadata: # [array] Fields - Fields to add to events referencing this token.
        - name: # [string] Name - Field name
          value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ---------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    splunkHecAPI: # [string] [required] Splunk HEC Endpoint - Absolute path on which to
listen for the Splunk HTTP Event Collector API requests. This input supports the /event
and /raw endpoints.
    metadata: # [array] Fields - Fields to add to every event. May be overridden by
fields added at the token or request level.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    allowedIndexes: # [array of strings] Allowed Indexes - List values allowed in HEC
event index field, allows wildcards. Leave blank to skip validation.
    splunkHecAcks: # [boolean] Splunk HEC Acks - Whether to enable Splunk HEC
acknowledgements
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
```

```
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # -------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # -------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_blob_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The storage account queue name blob notifications wil
be read from. Value must be a JavaScript expression (which can evaluate to a constant
value), enclosed in quotes or backticks. Can be evaluated only at init time. E.g.,
referencing a Global Variable: `myQueue-${C.vars.myVar}`
    fileFilter: # [string] Filename filter - Regex matching file names to download and
process. Defaults to: .*
    visibilityTimeout: # [number] Visibility timeout (secs) - The duration (in seconds)
that the received messages are hidden from subsequent retrieve requests after being
retrieved by a ReceiveMessage request.
    numReceivers: # [number] Num receivers - The Number of receiver processes to run, t
higher the number the better throughput at the expense of CPU overhead
    maxMessages: # [number] Max messages - The maximum number of messages to return in
poll request. Azure storage queues never returns more messages than this value (however
fewer messages might be returned). Valid values: 1 to 32.
    servicePeriodSecs: # [number] Service period (secs) - The duration (in seconds) whi
pollers should be validated and restarted if exited
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
```

```
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size for
each Parquet chunk.
    parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds) -
The maximum time to wait for a Parquet file's chunk to be downloaded. Processing will e
if a required chunk could not be downloaded within the time imposed by this setting.
    authType: # [string] Authentication method - Enter connection string directly, or
select a stored secret
    connectionString: # [string] Connection string - Enter your Azure Storage account
connection string. If left blank, Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

    # -------------- if authType is manual ---------------


    # --------------------------------------------------------

    textSecret: # [string] Connection string (text secret) - Select (or create) a store
text secret

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
```

```
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  elastic_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    elasticAPI: # [string] [required] Elasticsearch API endpoint - Absolute path on whi
to listen for Elasticsearch API requests. Defaults to /. _bulk will be appended
automatically, e.g., /myPath becomes /myPath/_bulk. Requests can then be made to either
/myPath/_bulk or /myPath/<myIndexName>/_bulk. Other entries are faked as success.
```

```
    authType: # [string] Authentication type - Elastic authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # --------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # --------------------------------------------------------


    # -------------- if authType is authTokens ---------------

    authTokens: # [array of strings] Token - Bearer tokens to include in the
authorization header

    # --------------------------------------------------------

    apiVersion: # [string] API Version - The API version to use for communicating with
the server.

    # -------------- if apiVersion is custom ---------------

    customAPIVersion: # [string] Custom API Version - Custom version information to
respond to requests

    # --------------------------------------------------------

    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    proxyMode: # [object]
      enabled: # [boolean] Enable Proxy Mode - Enable proxying of non-bulk API requests
to an external Elastic server. Enable this only if you understand the implications; see
docs for more details.

      # -------------- if enabled is true ---------------

      url: # [string] Proxy URL - URL of the Elastic server to proxy non-bulk requests
to, e.g., http://elastic:9200
      removeHeaders: # [array of strings] Remove headers - List of headers to remove fr
the request to proxy
      timeoutSec: # [number] Proxy request timeout - Amount of time, in seconds, to wai
for a proxy request to complete before aborting it.
      authType: # [string] Authentication method - Enter credentials directly, or selec
a stored secret

      # --------------------------------------------------------
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  confluent_cloud_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to use, eg.
yourAccount.confluent.cloud:9092
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not
authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the
system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
```

```
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in which
find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which to
find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    topics: # [array of strings] [required] Topic - Topic to subscribe to. Warning: To
optimize performance, Cribl suggests subscribing each Kafka Source to only a single
topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance belongs t
default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from earliest
available data, relevant only during initial subscription.
    kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
      disabled: # [boolean] Disabled - Enable Schema Registry

      # -------------- if disabled is false ---------------

      schemaRegistryURL: # [string] Schema Registry URL - URL for access to the Conflue
Schema Registry, i.e: http://localhost:8081
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are n
authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the
system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server Name
Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in whic
to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in which t
find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # ----------------------------------------------------------


      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for a
successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when connecting
```

```
brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable Authentication

      # -------------- if disabled is false ----------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

      # --------------------------------------------------------

    sessionTimeout: # [number] Session timeout (ms) -
      Timeout used to detect client failures when using Kafka's group management
facilities.
      If the client sends the broker no heartbeats before this timeout expires,
      the broker will remove this client from the group, and will initiate a rebalance.
      Value must be between the broker's configured group.min.session.timeout.ms and
group.max.session.timeout.ms.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_session.timeout.ms).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time for each worker to join the group after a rebalance has begu
      If the timeout is exceeded, the coordinator broker will remove the worker from th
group.
      See details [here]
(https://kafka.apache.org/documentation/#connectconfigs_rebalance.timeout.ms).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time between heartbeats to the consumer coordinator when using Kafka's
group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3 of t
sessionTimeout value.
      See details [here]
(https://kafka.apache.org/documentation/#consumerconfigs_heartbeat.interval.ms).
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ----------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
```

```
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  grafana_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ---------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    prometheusAPI: # [string] Remote Write API endpoint - Absolute path on which to
listen for Grafana Agent's Remote Write requests. Defaults to /api/prom/push, which wil
```

```
expand as: http://<yourâ¯💰upstreamâ¯💰URL>:<yourâ¯💰port>/api/prom/push.
    lokiAPI: # [string] Logs API endpoint - Absolute path on which to listen for Loki
logs requests. Defaults to /loki/api/v1/push, which will (in this example) expand as:
'http://<yourâ¯💰upstreamâ¯💰URL>:<yourâ¯💰port>/loki/api/v1/push'.
    keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait fo
additional data, after the last response was sent, before closing a socket connection.
This can be very useful when Grafana Agent remote write's request frequency is high so,
reusing connections, would help mitigating the cost of creating a new connection per
request. Note that Grafana Agent's embedded Prometheus would attempt to keep connection
open for up to 5 minutes.
    prometheusAuth: # [object]
      authType: # [string] Authentication type - Remote Write authentication type

      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for Basic authentication
      password: # [string] Password - Password for Basic authentication

      # --------------------------------------------------------


      # -------------- if authType is token ---------------

      token: # [string] Token - Bearer token to include in the authorization header

      # --------------------------------------------------------


      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret th
references your credentials

      # --------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

      # --------------------------------------------------------

    lokiAuth: # [object]
      authType: # [string] Authentication type - Loki logs authentication type

      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for Basic authentication
      password: # [string] Password - Password for Basic authentication

      # --------------------------------------------------------


      # -------------- if authType is token ---------------

      token: # [string] Token - Bearer token to include in the authorization header

      # --------------------------------------------------------
      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret th
references your credentials

      # --------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

      # --------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  loki_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    lokiAPI: # [string] [required] Logs API endpoint - Absolute path on which to listen
for Loki logs requests. Defaults to /loki/api/v1/push, which will (in this example)
expand as: 'http://<yourâ¯💰upstreamâ¯💰URL>:<yourâ¯💰port>/loki/api/v1/push'.
    authType: # [string] Authentication type - Loki logs authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # --------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # --------------------------------------------------------
    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # --------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text secre

    # --------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  prometheus_rw_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates tc
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # -------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    prometheusAPI: # [string] [required] Remote Write API endpoint - Absolute path on
which to listen for Prometheus requests. Defaults to /write, which will expand as:
http://<yourâ¯💰upstreamâ¯💰URL>:<yourâ¯💰port>/write.
    keepAliveTimeout: # [number] Keep alive timeout (seconds) - Maximum time to wait fc
additional data, after the last response was sent, before closing a socket connection.
This can be very useful when Prometheus remote write's request frequency is high so,
reusing connections, would help mitigating the cost of creating a new connection per
request. Note that Prometheus would attempt to keep connections open for up to 5 minute
    authType: # [string] Authentication type - Remote Write authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # -------------------------------------------------------
    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # -------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # -------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text secre

    # -------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # -------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
```

```
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  prometheus_input: # [object]
    dimensionList: # [array of strings] Extra Dimensions - Other dimensions to include
events
    discoveryType: # [string] Discovery Type - Target discovery mechanism. Use static t
manually enter a list of targets.

    # -------------- if discoveryType is static ---------------

    targetList: # [array of strings] Targets - List of Prometheus targets to pull metri
from. Values can be in URL or host[:port] format. For example:
http://localhost:9090/metrics, localhost:9090, or localhost. In cases where just
host[:port] is specified, the endpoint will resolve to 'http://host[:port]/metrics'.

    # ----------------------------------------------------------



    # -------------- if discoveryType is dns ---------------

    nameList: # [array] DNS Names - List of DNS names to resolve
    recordType: # [string] Record Type - DNS Record type to resolve
    scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting metri
    scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets

    # ----------------------------------------------------------



    # -------------- if discoveryType is ec2 ---------------

    usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
targets. Set to false if the private IP address should be used.
    scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting metri
    scrapePort: # [number] Metrics Port - The port number in the metrics URL for
discovered targets.
    scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets
    searchFilter: # [array] Search Filter - EC2 Instance Search Filter
      - Name: # [string] Filter Name - Search filter attribute name, see:
https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html for
more information. Attributes can be manually entered if not present in the drop down li
        Values: # [array of strings] Filter Values - Search Filter Values, if empty onl
"running" EC2 instances will be returned
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.
    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - Region where the EC2 is located
    endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to EC2-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signi
EC2 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rejec
```

```
certificates that cannot be verified against a valid CA (e.g., self-signed certificates
    enableAssumeRole: # [boolean] Enable for EC2 - Use Assume Role credentials to acces
EC2
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role t
assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming rol

    # ----------------------------------------------------------

    interval: # [number] Poll Interval - How often in minutes to scrape targets for
metrics, 60 must be evenly divisible by the value or save will fail.
    logLevel: # [string] [required] Log Level - Collector runtime Log Level
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should chec
in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Alive
Time periods before an inactive worker will have its job subscription revoked.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter credentials directly, or select
stored secret
    username: # [string] Username - Username for Prometheus Basic authentication
    password: # [string] Password - Password for Prometheus Basic authentication

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------
    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  office365_mgmt_input: # [object]
    type: # [string] Input Type
    tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
    appId: # [string] [required] App ID - Office 365 Azure Application ID
    timeout: # [number] Timeout (secs) - HTTP request inactivity timeout, use 0 to
disable
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should chec
in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Alive
Time periods before an inactive worker will have its job subscription revoked.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    planType: # [string] [required] Subscription Plan - Office 365 subscription plan fo
your organization, typically Enterprise
    publisherIdentifier: # [string] Publisher Identifier - Optional Publisher Identifie
to use in API requests, defaults to tenant id if not defined. For more information see
[here](https://docs.microsoft.com/en-us/office/office-365-management-api/office-365-
management-activity-api-reference#start-a-subscription)
    contentConfig: # [array] Content Types - Enable Office 365 Management Activity API
content types and polling intervals. Polling intervals are used to set up search date
range and cron schedule, e.g.: */${interval} * * * *. Because of this, intervals entere
must be evenly divisible by 60 to give a predictable schedule.
      - contentType: # [string] Content Type - Office 365 Management Activity API Conte
Type
        description: # [string] Interval Description - If interval type is minutes the
value entered must evenly divisible by 60 or save will fail
        interval: # [number] Interval
        logLevel: # [string] Log Level - Collector runtime Log Level
        enabled: # [boolean] Enabled
    authType: # [string] Authentication method - Enter client secret directly, or selec
a stored secret
    clientSecret: # [string] Client secret - Office 365 Azure client secret

    # -------------- if authType is manual ---------------
    # ----------------------------------------------------------

    textSecret: # [string] Client secret (text secret) - Select (or create) a stored te
secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  office365_service_input: # [object]
    type: # [string] Input Type
    tenantId: # [string] Tenant ID - Office 365 Azure Tenant ID
    appId: # [string] [required] App ID - Office 365 Azure Application ID
    timeout: # [number] Timeout (secs) - HTTP request inactivity timeout, use 0 to
disable
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should chec
```

```
in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Alive
Time periods before an inactive worker will have its job subscription revoked.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    contentConfig: # [array] Content Types - Enable Office 365 Service Communication AP
content types and polling intervals. Polling intervals are used to set up search date
range and cron schedule, e.g.: */${interval} * * * *. Because of this, intervals entere
for current and historical status must be evenly divisible by 60 to give a predictable
schedule.
      - contentType: # [string] Content Type - Office 365 Services API Content Type
        description: # [string] Interval Description - If interval type is minutes the
value entered must evenly divisible by 60 or save will fail
        interval: # [number] Interval
        logLevel: # [string] Log Level - Collector runtime Log Level
        enabled: # [boolean] Enabled
    authType: # [string] Authentication method - Enter client secret directly, or selec
a stored secret
    clientSecret: # [string] Client secret - Office 365 Azure client secret

    # -------------- if authType is manual ---------------


    # --------------------------------------------------------

    textSecret: # [string] Client secret (text secret) - Select (or create) a stored te
secret

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
```

```
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  office365_msg_trace_input: # [object]
    url: # [string] Report URL - URL to use when retrieving report data.
    interval: # [number] [required] Poll interval - How often (in minutes) to run the
report. Must divide evenly into 60 minutes to create a predictable schedule, or Save wi
fail.
    startDate: # [string] Date range start - Backward offset for the search range's hea
(E.g.: -3h@h) Message Trace data is delayed; this parameter (with Date range end)
compensates for delay and gaps.
    endDate: # [string] Date range end - Backward offset for the search range's tail.
(E.g.: -2h@h) Message Trace data is delayed; this parameter (with Date range start)
compensates for delay and gaps.
    logLevel: # [string] Log level - Log Level (verbosity) for collection runtime
behavior.
    timeout: # [number] Timeout (secs) - HTTP request inactivity timeout. Maximum is 24
(40 minutes); enter 0 to wait indefinitely.
    disableTimeFilter: # [boolean] Disable time filter - Disables time filtering of
events when a date range is specified.
    authType: # [string] Authentication method - Select authentication method.

    # -------------- if authType is manual ---------------

    username: # [string] Username - Username to run Message Trace API call.
    password: # [string] Password - Password to run Message Trace API call.

    # ----------------------------------------------------------


    # -------------- if authType is secret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials.

    # ----------------------------------------------------------


    # -------------- if authType is oauth ---------------

    clientSecret: # [string] Client secret - client_secret to pass in the OAuth request
parameter.
    tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Azure
Active Directory.
    clientId: # [string] Client ID - client_id to pass in the OAuth request parameter.
    resource: # [string] Resource - Resource to pass in the OAuth request parameter.

    # ----------------------------------------------------------


    # -------------- if authType is oauthSecret ---------------

    textSecret: # [string] Client secret - Select (or create) a secret that references
your client_secret to pass in the OAuth request parameter.
    tenantId: # [string] Tenant identifier - Directory ID (tenant identifier) in Azure
Active Directory.
    clientId: # [string] Client ID - client_id to pass in the OAuth request parameter.
    resource: # [string] Resource - Resource to pass in the OAuth request parameter.

    # ----------------------------------------------------------

    keepAliveTime: # [number] Keep Alive Time (seconds) - How often workers should chec
in with the scheduler to keep job subscription alive
    maxMissedKeepAlives: # [number] Worker Timeout (periods) - The number of Keep Alive
Time periods before an inactive worker will have its job subscription revoked.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
```

```
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  eventhub_input: # [object]
    type: # [string] Input Type
    brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to connect
to, e.g., yourdomain.servicebus.windows.net:9093. The hostname can be found in the host
portion of the primary or secondary connection string in Shared Access Policies.
    topics: # [array of strings] [required] Event Hub name - The name of the Event Hub
(a.k.a. Kafka topic) to subscribe to. Warning: To optimize performance, Cribl suggests
subscribing each Event Hubs Source to only a single topic.
    groupId: # [string] Group ID - Specifies the consumer group this instance belongs t
default is 'Cribl'.
    fromBeginning: # [boolean] From beginning - Whether to start reading from earliest
available data, relevant only during initial subscription.
    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for a
successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when connecting
brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable authentication.

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use. PLAI
is the only mechanism currently supported for Event Hubs Kafka brokers.
      username: # [string] Username - The username for authentication. For Event Hubs,
this should always be $ConnectionString.
      authType: # [string] Authentication method - Enter password directly, or select a
stored secret

      # ---------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this shou
always be false.

      # ---------------------------------------------------------

    sessionTimeout: # [number] Session timeout (ms) -
      Timeout (a.k.a session.timeout.ms in Kafka domain) used to detect client failures
when using Kafka's group management facilities.
      If the client sends the broker no heartbeats before this timeout expires, the
broker will remove this client from the group, and will initiate a rebalance.
      Value must be lower than rebalanceTimeout.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    rebalanceTimeout: # [number] Rebalance timeout (ms) -
      Maximum allowed time (a.k.a rebalance.timeout.ms in Kafka domain) for each worker
to join the group after a rebalance has begun.
```

```
      If the timeout is exceeded, the coordinator broker will remove the worker from th
group.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    heartbeatInterval: # [number] Heartbeat interval (ms) -
      Expected time (a.k.a heartbeat.interval.ms in Kafka domain) between heartbeats to
the consumer coordinator when using Kafka's group management facilities.
      Value must be lower than sessionTimeout, and typically should not exceed 1/3 of t
sessionTimeout value.
      See details [here](https://github.com/Azure/azure-event-hubs-for-
kafka/blob/master/CONFIGURATION.md).
    minimizeDuplicates: # [boolean] Minimize duplicates - Enable feature to minimize
duplicate events by only starting one consumer for each topic partition.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

```
  exec_input: # [object]
    disabled: # [boolean] Disabled - Enable/disable this input
    command: # [string] Command - Command to execute; supports Bourne shell syntax
    retries: # [number] Max retries - Maximum number of retry attempts in the event tha
the command fails.
    scheduleType: # [string] Schedule type - Select a schedule type; either an interval
(in seconds) or a cron-style schedule.

    # -------------- if scheduleType is interval ---------------

    interval: # [number] Interval - Interval between command executions in seconds.

    # ----------------------------------------------------------


    # -------------- if scheduleType is cronSchedule ---------------

    cronSchedule: # [string] Schedule - Cron schedule to execute the command on.

    # ----------------------------------------------------------

    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
```

```
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  firehose_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by any
client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
```

```
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  google_pubsub_input: # [object]
    type: # [string] Input Type
    topicName: # [string] Topic ID - ID of the topic to receive events from.
    subscriptionName: # [string] [required] Subscription ID - ID of the subscription to
use when receiving events.
    createTopic: # [boolean] Create topic - If enabled, create topic if it does not exi
    createSubscription: # [boolean] Create subscription - If enabled, create subscripti
if it does not exist

    # -------------- if createSubscription is true ---------------

    orderedDelivery: # [boolean] Ordered delivery - If enabled, receive events in the
order they were added to the queue. For this to work correctly, the process sending
events must have ordering enabled.

    # ---------------------------------------------------------

    region: # [string] Region - Region to retrieve messages from. Select 'default' to
allow Google to auto-select the nearest region. When using ordered delivery, the select
```

```
region must be allowed by message storage policy.
    googleAuthMethod: # [string] Authentication Method - Google authentication method.
Choose Auto to use environment variables PUBSUB_PROJECT and PUBSUB_CREDENTIALS.

    # -------------- if googleAuthMethod is manual ---------------

    serviceAccountCredentials: # [string] Service account credentials - Contents of
service account credentials (JSON keys) file downloaded from Google Cloud. To upload a
file, click the upload button at this field's upper right. As an alternative, you can u
environment variables (see [here](https://googleapis.dev/ruby/google-cloud-
pubsub/latest/file.AUTHENTICATION.html)).

    # ----------------------------------------------------------


    # -------------- if googleAuthMethod is secret ---------------

    secret: # [string] Service account credentials (text secret) - Select (or create) a
stored text secret

    # ----------------------------------------------------------

    maxBacklog: # [number] Max backlog - If Destination exerts backpressure, this setti
limits how many inbound events Stream will queue for processing before it stops
retrieving events.
    requestTimeout: # [number] Request timeout (ms) - Pull request timeout, in
milliseconds.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
```

```
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  cribl_input: # [object]
    type: # [string] Input Type
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------------
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  cribl_tcp_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ---------------------------------------------------------

    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------
    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  cribl_http_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by any
client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
```

```
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  tcpjson_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------
      certificateName: # [string] Certificate name - The name of the predefined certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require clients to present their certificates. Used to perform client authentication using SSL certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from connections.

      # ----------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter a token directly, or provide a secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client (in authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations, optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  system_metrics_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive metri
collections.
    host: # [object]
      mode: # [string]  - Select level of detail for host metrics

      # -------------- if mode is custom ---------------

      custom: # [object]
        system: # [object]
          mode: # [string]  - Select the level of details for system metrics

          # -------------- if mode is custom ---------------

          processes: # [boolean] Process metrics - Generate metrics for the numbers of
processes in various states

          # --------------------------------------------------------

        cpu: # [object]
          mode: # [string]  - Select the level of details for CPU metrics

          # -------------- if mode is custom ---------------

          perCpu: # [boolean] Per CPU metrics - Generate metrics for each CPU
          detail: # [boolean] Detailed metrics - Generate metrics for all CPU states
          time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time counter

          # --------------------------------------------------------

        memory: # [object]
```

```
          mode: # [string]  - Select the level of details for memory metrics

          # -------------- if mode is custom ---------------

          detail: # [boolean] Detailed metrics - Generate metrics for all memory states

          # --------------------------------------------------------

        network: # [object]
          mode: # [string]  - Select the level of details for network metrics

          # -------------- if mode is custom ---------------

          devices: # [array of strings] Interface filter - Network interfaces to
include/exclude. E.g.: eth0, !lo, etc. All interfaces are included if this list is empt
          perInterface: # [boolean] Per interface metrics - Generate separate metrics f
each interface
          detail: # [boolean] Detailed metrics - Generate full network metrics

          # --------------------------------------------------------

        disk: # [object]
          mode: # [string]  - Select the level of details for disk metrics

          # -------------- if mode is custom ---------------

          devices: # [array of strings] Device filter - Block devices to include/exclud
E.g.: sda*, !loop*, etc. Wildcards and ! (not) operators are supported. All devices are
included if this list is empty.
          mountpoints: # [array of strings] Mountpoint filter - Filesystem mountpoints
include/exclude. E.g.: /, /home, !/proc*, !/tmp, etc. Wildcards and ! (not) operators a
supported. All mountpoints are included if this list is empty.
          fstypes: # [array of strings] Filesystem type filter - Filesystem types to
include/exclude. E.g.: ext4, !*tmpfs, !squashfs, etc. Wildcards and ! (not) operators a
supported. All types are included if this list is empty.
          perDevice: # [boolean] Per device metrics - Generate separate metrics for eac
device
          detail: # [boolean] Detailed metrics - Generate full disk metrics

          # --------------------------------------------------------


      # --------------------------------------------------------

    container: # [object]
      mode: # [string]  - Select the level of detail for container metrics

      # -------------- if mode is basic ---------------

      dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNIX-
domain socket
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker AP

      # --------------------------------------------------------


      # -------------- if mode is all ---------------

      dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNIX-
domain socket
```

```
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker AP

      # ----------------------------------------------------------


      # -------------- if mode is custom ---------------

      filters: # [array] Container Filters - Containers matching any of these will be
included. All are included if this is empty.
        - expr: # [string] Expression
      allContainers: # [boolean] All containers - Include stopped and paused containers
      perDevice: # [boolean] Per device metrics - Generate separate metrics for each
device
      detail: # [boolean] Detailed metrics - Generate full container metrics
      dockerSocket: # [array of strings] Docker socket - Full paths for Docker's UNIX-
domain socket
      dockerTimeout: # [number] Docker timeout - Timeout, in seconds, for the Docker AP

      # ----------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    persistence: # [object] persistence
      enable: # [boolean] Enable disk persistence - Persist metrics on disk

      # -------------- if enable is true ---------------

      timeWindow: # [string] Bucket time span - Time span for each file bucket
      maxDataSize: # [string] Max data size - Maximum disk space allowed to be consumed
(e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data (e.g
2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`

      # ----------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue
    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  system_state_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive state
collections.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    collectors: # [object]
      hostsfile: # [object]
        enable: # [boolean] Enabled
    persistence: # [object]
      enable: # [boolean] Enable disk persistence - Persist metrics on disk

      # -------------- if enable is true ---------------

      timeWindow: # [string] Bucket time span - Time span for each file bucket
      maxDataSize: # [string] Max data size - Maximum disk space allowed to be consumed
(e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data (e.g
2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`

      # ---------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  windows_metrics_input: # [object]
    interval: # [number] Polling interval - Time, in seconds, between consecutive metri
collections.
    host: # [object]
      mode: # [string]  - Select level of detail for host metrics

      # -------------- if mode is custom ---------------

      custom: # [object]
        system: # [object]
          mode: # [string]  - Select the level of details for system metrics

          # -------------- if mode is custom ---------------

          detail: # [boolean] Detailed metrics - Generate metrics for all system
information

          # ---------------------------------------------------------

        cpu: # [object]
          mode: # [string]  - Select the level of details for CPU metrics

          # -------------- if mode is custom ---------------

          perCpu: # [boolean] Per CPU metrics - Generate metrics for each CPU
          detail: # [boolean] Detailed metrics - Generate metrics for all CPU states
          time: # [boolean] CPU time metrics - Generate raw, monotonic CPU time counter

          # ----------------------------------------------------------

        memory: # [object]
          mode: # [string]  - Select the level of details for memory metrics

          # -------------- if mode is custom ---------------

          detail: # [boolean] Detailed metrics - Generate metrics for all memory states

          # ----------------------------------------------------------

        network: # [object]
          mode: # [string]  - Select the level of details for network metrics

          # -------------- if mode is custom ---------------

          devices: # [array of strings] Interface filter - Network interfaces to
include/exclude. All interfaces are included if this list is empty.
          perInterface: # [boolean] Per interface metrics - Generate separate metrics f
each interface
          detail: # [boolean] Detailed metrics - Generate full network metrics

          # ----------------------------------------------------------

        disk: # [object]
          mode: # [string]  - Select the level of details for disk metrics

          # -------------- if mode is custom ---------------

          volumes: # [array of strings] Volume filter - Windows volumes to
include/exclude. E.g.: C:, !E:, etc. Wildcards and ! (not) operators are supported. All
volumes are included if this list is empty.
          perVolume: # [boolean] Per volume metrics - Generate separate metrics for eac
volume

          # ----------------------------------------------------------


      # ----------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    persistence: # [object] persistence
      enable: # [boolean] Enable disk persistence - Persist metrics on disk

      # -------------- if enable is true ---------------

      timeWindow: # [string] Bucket time span - Time span for each file bucket
      maxDataSize: # [string] Max data size - Maximum disk space allowed to be consumed
(e.g., 420MB or 4GB). Once reached, older data will be deleted.
      maxDataTime: # [string] Max data age - Maximum amount of time to retain data (e.g
2h or 4d). Once reached, older data will be deleted.
      compress: # [string] Compression - Select data compression format. Optional.
      destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`
```

```
      # ---------------------------------------------------------

    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  crowdstrike_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read
notifications from. When a non-AWS URL is specified, format must be: '{url}/myQueueName
E.g., 'https://host:port/myQueueName'. Value must be a JavaScript expression (which can
evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only a
init time. E.g., referencing a Global Variable:
`https://host:port/myQueue-${C.vars.myVar}`.
    fileFilter: # [string] Filename filter - Regex matching file names to download and
process. Defaults to: .*
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Leave
empty if SQS queue is in same AWS account.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.
```

```
    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # ---------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # ---------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the S3 bucket and SQS queue are locate
Required, unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signi
S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rejec
certificates that cannot be verified against a valid CA (e.g., self-signed certificates
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility timeout seconds - The duration (in seconds
that received messages are hidden from subsequent retrieve requests, after being
retrieved by a ReceiveMessage request. This value also automatically extends this
duration, to prevent it from expiring before processing completes.
    numReceivers: # [number] Num receivers - The Number of receiver processes to run, t
higher the number the better throughput at the expense of CPU overhead
    socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in seconds).
Increase this value if timeouts occur due to backpressure.
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to access
S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role t
assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming rol
    enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials when
accessing SQS.
    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for events
before trying polling again. The lower the number the higher the AWS bill. The higher t
number the longer it will take for the source to react to configuration changes and
system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ---------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ---------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  datadog_agent_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
```

```
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ----------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each incomin
metric to multiple events, one per data point. This works well when sending metrics to
statsd-type output. If sending metrics to DatadogHQ or any destination that accepts
arbitrary JSON, leave toggled to No (the default).
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    proxyMode: # [object]
      enabled: # [boolean] Forward API key validation requests - Toggle to Yes to send
key validation requests from Datadog Agent to the Datadog API. If toggled to No (the
default), Stream handles key validation requests by always responding that the key is
valid.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------
    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  datagen_input: # [object]
    samples: # [array] Datagen - List of datagens
      - sample: # [string] Data Generator File - Name of the datagen file
        eventsPerSec: # [number] Events Per Second Per Worker Node - Maximum no. of
events to generate per second per worker node. Defaults to 10.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
```

```
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  http_raw_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    authTokens: # [array of strings] Auth tokens - Shared secrets to be provided by any
client (Authorization: <token>). If empty, unauthed access is permitted.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # --------------------------------------------------------

    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
```

```
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    allowedPaths: # [array of strings] Allowed URI paths - List of URI paths accepted b
this input, wildcards are supported, e.g /api/v*/hook. Defaults to allow all.
    allowedMethods: # [array of strings] Allowed HTTP methods - List of HTTP methods
accepted by this input, wildcards are supported, e.g. P*, GET. Defaults to allow all.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  kinesis_input: # [object]
    type: # [string] Input Type
    streamName: # [string] Stream name - Kinesis stream name to read data from.
    serviceInterval: # [number] Service Period - Time interval in minutes between
consecutive service calls
    shardExpr: # [string] Shard selection expression - A JS expression to be called wit
each shardId for the stream, if the expression evalutates to a truthy value the shard
```

```
will be processed.
    shardIteratorType: # [string] Shard iterator start - Location at which to start
reading a shard for the first time.
    payloadFormat: # [string] Record data format - Format of data inside the Kinesis
Stream records. Gzip compression is automatically detected.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # ---------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # ---------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is located
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defaults
to AWS' Region-specific endpoint. Otherwise, it must point to Kinesis stream-compatible
endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signi
Kinesis stream requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rejec
certificates that cannot be verified against a valid CA (e.g., self-signed certificates
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role credentia
to access Kinesis stream
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role t
assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming rol
    verifyKPLCheckSums: # [boolean] Verify KPL checksums - Verify Kinesis Producer
Library (KPL) event checksums
    avoidDuplicates: # [boolean] Avoid duplicate records - Yes means: when resuming
streaming from a stored state, Stream will read the next available record, rather than
rereading the last-read record. Enabling this can cause data loss after a Worker Node's
unexpected shutdown or restart.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.
```

```
  # ----------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
   mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
   maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
   commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
   maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
   maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
   path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
   compress: # [string] Compression - Codec to use to compress the persisted data.

  # ----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
 logstream_input: # [object]
  type: # [string] Input Type
  disabled: # [boolean] Disabled - Enable/disable this input
  host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
  port: # [number] [required] Port - Port to listen to.
  tls: # [object] TLS settings (server side)
   disabled: # [boolean] Disabled

   # -------------- if disabled is false ---------------

   certificateName: # [string] Certificate name - The name of the predefined
certificate.
   privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
   passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
   certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
   caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
   requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
   minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
   maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

   # ----------------------------------------------------------
```

```
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    authType: # [string] Authentication method - Enter a token directly, or provide a
secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client (in
authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------


    # --------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # --------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
```

```
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  criblmetrics_input: # [object]
    type: # [string] Input Type
    prefix: # [string] Metric Name Prefix - A prefix that is applied to the metrics
provided by Cribl Stream
    fullFidelity: # [boolean] Full Fidelity - Include granular metrics.  Disabling this
will drop the following metrics events: `cribl.logstream.host.
(in_bytes,in_events,out_bytes,out_events)`, `cribl.logstream.index.
(in_bytes,in_events,out_bytes,out_events)`, `cribl.logstream.source.
(in_bytes,in_events,out_bytes,out_events)`, `cribl.logstream.sourcetype.
(in_bytes,in_events,out_bytes,out_events)`.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
```

```
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  metrics_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    udpPort: # [number] UDP Port - Enter UDP port number to listen on. Not required if
listening on TCP.
    tcpPort: # [number] TCP Port - Enter TCP port number to listen on. Not required if
listening on UDP.
    maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
buffer when downstream is blocking. Only applies to UDP.
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to send data
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ----------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
```

```
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  s3_input: # [object]
    type: # [string] Input Type
    queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read
notifications from. When a non-AWS URL is specified, format must be: '{url}/myQueueName
E.g., 'https://host:port/myQueueName'. Value must be a JavaScript expression (which can
evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only a
init time. E.g., referencing a Global Variable:
`https://host:port/myQueue-${C.vars.myVar}`.
    fileFilter: # [string] Filename filter - Regex matching file names to download and
process. Defaults to: .*
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Leave
empty if SQS queue is in same AWS account.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # ----------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # ----------------------------------------------------------
    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the S3 bucket and SQS queue are locate
Required, unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signi
S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rejec
certificates that cannot be verified against a valid CA (e.g., self-signed certificates
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility timeout seconds - The duration (in seconds
that received messages are hidden from subsequent retrieve requests, after being
retrieved by a ReceiveMessage request. This value also automatically extends this
duration, to prevent it from expiring before processing completes.
    numReceivers: # [number] Num receivers - The Number of receiver processes to run, t
higher the number the better throughput at the expense of CPU overhead
    socketTimeout: # [number] Socket timeout - Socket inactivity timeout (in seconds).
Increase this value if timeouts occur due to backpressure.
    skipOnError: # [boolean] Skip file on error - Toggle to Yes to skip files that
trigger a processing error. Defaults to No, which enables retries after processing
errors.
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to access
S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role t
assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming rol
    enableSQSAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials when
accessing SQS.
    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    parquetChunkSizeMB: # [number] Max Parquet chunk size (MB) - Maximum file size for
each Parquet chunk.
    parquetChunkDownloadTimeout: # [number] Parquet chunk download timeout (seconds) -
The maximum time to wait for a Parquet file's chunk to be downloaded. Processing will e
if a required chunk could not be downloaded within the time imposed by this setting.
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for events
before trying polling again. The lower the number the higher the AWS bill. The higher t
number the longer it will take for the source to react to configuration changes and
system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
```

```
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  snmp_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    port: # [number] [required] UDP Port - UDP port to receive SNMP traps on. Defaults
162.
    maxBufferSize: # [number] Max Buffer Size (events) - Maximum number of events to
buffer when downstream is blocking.
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to send data
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------
    connections: # [array] Quick Connections - Direct connections to Destinations, 
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a 
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem 
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold 
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send 
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue 
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback 
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in 
Stream.
  open_telemetry_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined 
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private 
key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require 
clients to present their certificates. Used to perform client authentication using SSL 
certs.

      # ----------------------------------------------------------
    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    activityLogSampleRate: # [number] Activity log sample rate - How often request
activity is logged at the `info` level. A value of 1 would log every request, 10 every
10th request, etc.
    requestTimeout: # [number] Request timeout (seconds) - How long to wait for an
incoming request to complete before aborting it. Use 0 to disable.
    extractSpans: # [boolean] Extract spans - Toggle to Yes to extract each incoming sp
to a separate event.
    extractMetrics: # [boolean] Extract metrics - Toggle to Yes to extract each incomin
Gauge or IntGauge metric to multiple events, one per data point.
    maxActiveCxn: # [number] Max active connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    authType: # [string] Authentication type - OpenTelemetry authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # --------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # --------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that
references your credentials

    # --------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text secre

    # --------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
   - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
    output: # [string] Destination - Select a Destination.

  # -----------------------------------------------------------

  environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  pqEnabled: # [boolean] Enable Persistent Queue

  # -------------- if pqEnabled is true ---------------

  pq: # [object]
   mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
   maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
   commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
   maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
   maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
   path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
   compress: # [string] Compression - Codec to use to compress the persisted data.

  # -----------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
 sqs_input: # [object]
  type: # [string] Input Type
  queueName: # [string] Queue - The name, URL, or ARN of the SQS queue to read events
from. When a non-AWS URL is specified, format must be: '{url}/myQueueName'. E.g.,
'https://host:port/myQueueName'. Value must be a JavaScript expression (which can
evaluate to a constant value), enclosed in quotes or backticks. Can only be evaluated a
init time. E.g. referencing a Global Variable:
`https://host:port/myQueue-${C.vars.myVar}`.
  queueType: # [string] [required] Queue Type - The queue type used (or created).
Defaults to Standard

  # -------------- if queueType is standard ---------------

  numReceivers: # [number] Num receivers - The Number of receiver processes to run, t
higher the number the better throughput at the expense of CPU overhead

  # -----------------------------------------------------------

  awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Leave
empty if SQS queue is in same AWS account.
  createQueue: # [boolean] Create Queue - Create queue if it does not exist.
  awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method. Choose Auto to use IAM roles.

  # -------------- if awsAuthenticationMethod is manual ---------------
```

```
    awsApiKey: # [string] Access key - Access key

    # ----------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # ----------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the SQS queue is located. Required,
unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to SQS-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signi
SQS requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to rejec
certificates that cannot be verified against a valid CA (e.g., self-signed certificates
    enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to acces
SQS
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role t
assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming rol
    maxMessages: # [number] Max Messages - The maximum number of messages SQS should
return in a poll request. Amazon SQS never returns more messages than this value
(however, fewer messages might be returned). Valid values: 1 to 10.
    visibilityTimeout: # [number] Visibility Timeout Seconds - The duration (in seconds
that the received messages are hidden from subsequent retrieve requests after being
retrieved by a ReceiveMessage request.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pollTimeout: # [number] Poll timeout (secs) - The amount of time to wait for events
before trying polling again. The lower the number the higher the AWS bill. The higher t
number the longer it will take for the source to react to configuration changes and
system restarts.
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue
    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  syslog_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. For IPv4 (all addresses), use the
default '0.0.0.0'. For IPv6, enter '::' (all addresses) or specify an IP address.
    udpPort: # [number] UDP port - Enter UDP port number to listen on. Not required if
listening on TCP.
    tcpPort: # [number] TCP port - Enter TCP port number to listen on. Not required if
listening on UDP.
    maxBufferSize: # [number] Max buffer size (events) - Maximum number of events to
buffer when downstream is blocking. Only applies to UDP.
    ipWhitelistRegex: # [string] IP allowlist regex - Regex matching IP addresses that
are allowed to send data
    timestampTimezone: # [string] Default timezone - Timezone to assign to timestamps
without timezone info.
    singleMsgUdpPackets: # [boolean] Single msg per UDP - Whether to treat UDP packet
data received as full syslog message
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2
    keepFieldsList: # [array of strings] Fields to keep - Wildcard list of fields to ke
from source data, * = ALL (default)
    octetCounting: # [boolean] Octet count framing - Enable if incoming messages use
octet counting per RFC 6587.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
```

```
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ----------------------------------------------------------

    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  file_input: # [object]
    mode: # [string]  - Choose how to discover files to monitor.
    # -------------- if mode is manual ---------------

    path: # [string] Search path - Directory path to search for files. Environment
variables will be resolved, e.g. $CRIBL_HOME/log/.
    depth: # [number] Max depth - Set how many subdirectories deep to search. Use 0 to
search only files in the given path, 1 to also look in its immediate subdirectories, et
Leave it empty for unlimited depth.

    # ----------------------------------------------------------

    interval: # [number] Polling interval - Time, in seconds, between scanning for file
    filenames: # [array of strings] Filename allowlist - The full path of discovered
files are matched against this wildcard list.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
```

```
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # --------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  tcp_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates tc
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # --------------------------------------------------------

    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    enableHeader: # [boolean] Enable Header - If enabled, client will pass the header
record with every new connection. The header can contain an authToken, and an object wi
a list of fields and values to add to every event. These fields can be used to simplify
Event Breaker selection, routing, etc. Header has this format, and must be followed by
newline: { "authToken" : "myToken", "fields": { "field1": "value1", "field2": "value2"
}

    # -------------- if enableHeader is true ----------------
    authType: # [string] Authentication method - Enter a token directly, or provide a
secret referencing a token

    # ----------------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and
process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  appscope_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    ipWhitelistRegex: # [string] IP Allowlist Regex - Regex matching IP addresses that
are allowed to establish a connection.
    maxActiveCxn: # [number] Max Active Connections - Maximum number of active
connections allowed per Worker Process, use 0 for unlimited
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
```

```
proxied by a device that supports Proxy Protocol V1 or V2.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event
breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (
milliseconds) the Event Breaker will wait for new data to be sent to a specific channel
before flushing the data stream out, as-is, to the Pipelines.
    enableUnixPath: # [boolean] UNIX domain socket - Toggle to Yes to specify a file-
backed UNIX domain socket connection, instead of a network host and port.

    # -------------- if enableUnixPath is false ---------------

    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] Port - Port to listen to.
    tls: # [object] TLS settings (server side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] Certificate path - Path on server containing certificates to
use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] CA certificate path - Path on server containing CA certificate
to use. PEM format. Can reference $ENV_VARS.
      requestCert: # [boolean] Authenticate client (mutual auth) - Whether to require
clients to present their certificates. Used to perform client authentication using SSL
certs.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections.

      # ---------------------------------------------------------


    # ---------------------------------------------------------


    # -------------- if enableUnixPath is true ---------------

    unixSocketPath: # [string] UNIX socket path - Path to the UNIX domain socket to
listen on.
    unixSocketPerms: # [string,number] UNIX socket permissions - Permissions to set for
socket e.g., 777. If empty, falls back to the runtime user's default permissions.

    # ---------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide a
secret referencing a token
    authToken: # [string] Auth token - Shared secret to be provided by any client (in
authToken header field). If empty, unauthed access is permitted.

    # -------------- if authType is manual ---------------
    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  wef_input: # [object]
    type: # [string] Input Type
    disabled: # [boolean] Disabled - Enable/disable this input
    host: # [string] Address - Address to bind on. Defaults to 0.0.0.0 (all addresses).
    port: # [number] [required] Port - Port to listen to.
    tls: # [object] [required] mTLS settings
```

```
      disabled: # [boolean] Disabled - Enable TLS
      rejectUnauthorized: # [boolean] Validate client certs - Required for WEF
certificate authentication.
      requestCert: # [boolean] Authenticate client - Required for WEF certificate
authentication.
      certificateName: # [string] Certificate name - Name of the predefined certificate
      privKeyPath: # [string] Private key path - Path on server containing the private
key to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      certPath: # [string] [required] Certificate path - Path on server containing
certificates to use. PEM format. Can reference $ENV_VARS.
      caPath: # [string] [required] CA certificate path - Path on server containing CA
certificates to use. PEM format. Can reference $ENV_VARS.
      commonNameRegex: # [string] Common name - Regex matching allowable common names i
peer certificates' subject attribute.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to accept from
connections.
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to accept from
connections
    maxActiveReq: # [number] Max active requests - Maximum number of active requests pe
Worker Process. Use 0 for unlimited.
    enableProxyHeader: # [boolean] Enable proxy protocol - Enable if the connection is
proxied by a device that supports Proxy Protocol V1 or V2.
    captureHeaders: # [boolean] Capture request headers - Toggle this to Yes to add
request headers to events, in the __headers field.
    caFingerprint: # [string] CA fingerprint override - SHA1 fingerprint expected by th
client, if it does not match the first certificate in the configured CA chain
    allowMachineIdMismatch: # [boolean] Allow MachineID mismatch - Allow events to be
ingested even if their MachineID does not match the client certificate CN.
    subscriptions: # [array] [required] Subscriptions - Subscriptions to events on
forwarding endpoints.
      - subscriptionName: # [string] Name - Friendly name for this subscription.
        version: # [string] Version - Version UUID for this subscription. If any
subscription parameters are modified, this value will change.
        contentFormat: # [string] Format - Content format in which the endpoint should
deliver events.
        heartbeatInterval: # [number] Heartbeat - Max time (in seconds) between endpoin
checkins before considering it unavailable.
        batchTimeout: # [number] Batch timeout - Interval (in seconds) over which the
endpoint should collect events before sending them to Stream.
        readExistingEvents: # [boolean] Read existing events - Set to Yes if a newly-
subscribed endpoint should send previously existing events. Set to No to only receive n
events
        sendBookmarks: # [boolean] Use bookmarks - If toggled to Yes, @{product} will
keep track of which events have been received, resuming from that point after a re-
subscription. This setting takes precedence over 'Read existing events' -- see the
documentation for details.
        compress: # [boolean] Compression - If toggled to Yes, Stream will receive
compressed events from the source.
        targets: # [array of strings] Targets - Enter the DNS names of the endpoints th
should forward these events. You may use wildcards, for example: *.mydomain.com
        querySelector: # [string] Query builder mode - Select the query builder mode.
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.

    # -----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # -----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  win_event_logs_input: # [object]
    type: # [string] Input Type
    logNames: # [array of strings] Event Logs - Enter the event logs to collect. Run
"Get-WinEvent -ListLog *" in PowerShell to see the available logs.
    readMode: # [string] Read Mode - Read all stored and future event logs, or only
future events.
    interval: # [number] Polling Interval - Time, in seconds, between checking for new
entries.
    batchSize: # [number] Batch Size - Maximum number of event records to read in one
interval.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    disabled: # [boolean] Disabled - Enable/disable this input
    pipeline: # [string] Pipeline - Pipeline to process data from this Source before
sending it through the Routes.
    sendToRoutes: # [boolean]  - Select whether to send data to Routes, or directly to
Destinations.

    # -------------- if sendToRoutes is false ---------------

    connections: # [array] Quick Connections - Direct connections to Destinations,
optionally via a Pipeline or a Pack.
      - pipeline: # [string] Pipeline/Pack - Select Pipeline or Pack. Optional.
        output: # [string] Destination - Select a Destination.
    # ----------------------------------------------------------

    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    pqEnabled: # [boolean] Enable Persistent Queue

    # -------------- if pqEnabled is true ---------------

    pq: # [object]
      mode: # [string] Mode - With Smart mode, PQ will write events to the filesystem
only when it detects backpressure from the processing engine. WithÂ AlwaysÂ On mode, PQ
will always write events directly to the queue before forwarding them to the processing
engine.
      maxBufferSize: # [number] Max buffer size - The maximum amount of events to hold
in-memory before dumping the events to disk.
      commitFrequency: # [number] Commit frequency - The number of events to send
downstream before committing that Stream has read them.
      maxFileSize: # [string] Max file size - The maximum size to store in each queue
file before closing and optionally compressing (KB, MB, etc.).
      maxSize: # [string] Max queue size - The maximum amount of disk space the queue i
allowed to consume. Once reached, the system stops queueing and applies the fallback
Queue-full behavior. Enter a numeral with units of KB, MB, etc.
      path: # [string] Queue file path - The location for the persistent queue files. T
this field's value, the system will append: /<worker-id>/inputs/<input-id>.
      compress: # [string] Compression - Codec to use to compress the persisted data.

    # ----------------------------------------------------------

    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

;

10.6.6. instance.yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml`.

$CRIBL_HOME/local/_system/instance.yml

```
distributed:
    # mode master | worker | single | edge | managed-edge
  mode: master
  master:
    host: 0.0.0.0
    port: 4203
    tls:
      disabled: true
    ipWhitelistRegex: /.&ast;/
    maxActiveCxn: 0
    authToken: criblmaster
    enabledWorkerRemoteAccess: true
    compression: none
    connectionTimeout: 5000
    writeTimeout: 10000
  group: default
  envRegex: /^CRIBL_/
  tags:
       - tag1
       - tag2
       - tag42
```

;

10.6.7. 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 run
by the same worker node.
  collector: # [object]
    type: # [string] Collector type - The type of collector to run.

    # -------------- if type is azure_blob ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined
Destination that will be used to auto-populate collector settings.
      authType: # [string] Authentication method - Enter authentication data
directly, or select a secret referencing your auth data

      # -------------- if authType is manual ---------------

      connectionString: # [string] Connection string - Enter your Azure storage
account Connection String. If left blank, Cribl Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

      # --------------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Connection string (text secret) - Text secret

      # --------------------------------------------------------

      containerName: # [string] Container name - Name of the container to collect
from. This value can be a constant or a JavaScript expression that can only be
evaluated at init time. E.g. referencing a Global Variable:
`myBucket-${C.vars.myVar}`.
      path: # [string] Path - The directory from which to collect data. Templating
is supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also
supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as
context for expressions that enrich discovery results. E.g.: given a template
/path/${epoch}, an extractor under key "epoch" with an expression {date: new
Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives
token under "value" variable, and evaluates to populate event fields, e.g.: {date:
new Date(+value*1000)}
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
objects to batch before recording as results.

    # --------------------------------------------------------


    # -------------- if type is filesystem ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - Select a predefined configuration
(e.g., a Destination) to auto-populate collector settings.
      path: # [string] Directory - The directory from which to collect data.
Templating is supported, e.g.: /myDir/${datacenter}/${host}/${app}/. Time-based
tokens are also supported, e.g.: /myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
```

```
      extractors: # [array] Path extractors - Allows using template tokens as context for expressions that enrich discovery results. E.g.: given a template /path/${epoch}, an extractor under key "epoch" with an expression {date: new Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template directory, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives token under "value" variable, and evaluates to populate event fields, e.g.: {date: new Date(+value*1000)}
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (files) - Maximum number of metadata files to batch before recording as results.

    # ----------------------------------------------------------


    # -------------- if type is google_cloud_storage ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined Destination that will be used to auto-populate collector settings.
      bucket: # [string] Bucket name - Name of the bucket to collect from. This value can be a constant or a JavaScript expression that can only be evaluated at init time. E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
      path: # [string] Path - The directory from which to collect data. Templating is supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as context for expressions that enrich discovery results. E.g.: given a template /path/${epoch}, an extractor under key "epoch" with an expression {date: new Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives token under "value" variable, and evaluates to populate event fields, e.g.: {date: new Date(+value*1000)}
      endpoint: # [string] Endpoint - Google Cloud Storage service endpoint. If empty, the endpoint will be automatically constructed using service credentials.
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector event time filtering when a date range is specified.
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata objects to batch before recording as results.
      authType: # [string] Authentication method - Enter account credentials manually, or select a secret that references your credentials.

      # -------------- if authType is manual ---------------

      serviceAccountCredentials: # [string] Service account credentials - Contents of Google Cloud service account credentials (JSON keys) file. To upload a file, click the upload button at this field's upper right.

      # ----------------------------------------------------------


      # -------------- if authType is secret ---------------

      textSecret: # [string] Service account credentials (text secret) - Select (or create) a stored text secret that references your credentials.

      # ----------------------------------------------------------
    # ----------------------------------------------------------


    # -------------- if type is health_check ---------------

    conf: # [object] [required]
      discovery: # [object]
        discoverType: # [string] Discover Type - Defines how task discovery will be
performed. Use None to skip the discovery. Use HTTP Request to make a REST call to
discover tasks. Use Item List to enumerate items for collect to retrieve. Use JSON
Response to manually define discover tasks as a JSON array of objects. Each entry
returned by the discover operation will result in a collect task.

        # -------------- if discoverType is http ---------------

        discoverUrl: # [string] Discover URL - Expression to derive URL to use for
the Discover operation (can be a constant).
        discoverMethod: # [string] Discover method - Discover HTTP method.
        discoverRequestHeaders: # [array] Discover Headers - Optional discover
request headers.
          - name: # [string] Name - Header name.
            value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).
        discoverDataField: # [string] Discover Data Field - Path to field in the
response object which contains discover results (e.g.: level1.name), leave blank if
the result is an array.

        # ----------------------------------------------------------


        # -------------- if discoverType is json ---------------

        manualDiscoverResult: # [string] Discover result - Allows hard-coding the
Discover result. Must be a JSON object. Works with the Discover Data field.
        discoverDataField: # [string] Discover data field - Within the response
JSON, name of the field or array element to pull results from. LeaveÂ blank if the
result is an array of values. Sample entry: items, json: { items: [{id: 'first'},
{id: 'second'}] }

        # ----------------------------------------------------------


        # -------------- if discoverType is list ---------------

        itemList: # [array of strings] Discover items - Comma-separated list of
items to return from the Discover task. Each item returned will generate a collect
task, and can be referenced using `${id}` in the collect URL, headers, or
parameters.

        # ----------------------------------------------------------

      collectUrl: # [string] Health check URL - Expression to derive URL to use for
the health check operation (can be a constant).
      collectMethod: # [string] [required] Health check method - Health check HTTP
method.

      # -------------- if collectMethod is get ---------------

      collectRequestParams: # [array] Health check parameters - Optional health
```

```
check request parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter
value (can be a constant).

      # ----------------------------------------------------------


      # -------------- if collectMethod is post ---------------

      collectRequestParams: # [array] Health check parameters - Optional health
check request parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter
value (can be a constant).

      # ----------------------------------------------------------


      # -------------- if collectMethod is post_with_body ---------------

      collectBody: # [string] Health check POST Body - Template for POST body to
send with the health check request. You can reference parameters from the Discover
response, using template params of the form: ${variable}.

      # ----------------------------------------------------------

      collectRequestHeaders: # [array] Health check headers - Optional health check
request headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).
      authenticateCollect: # [boolean] Authenticate health check - Enable to make
auth health check call.
      authentication: # [string] [required] Authentication - Authentication method
for Discover and Collect REST calls. You can specify API Keyâ¯💳based authentication
by adding the appropriate CollectÂ headers.

      # -------------- if authentication is basic ---------------

      username: # [string] Username - Basic authentication username
      password: # [string] Password - Basic authentication password

      # ----------------------------------------------------------


      # -------------- if authentication is basicSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your credentials

      # ----------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication Headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).

      # ----------------------------------------------------------


      # -------------- if authentication is loginSecret ----------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is
expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your login credentials
      loginBody: # [string] POST Body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication Headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header
value (can be a constant).

      # ----------------------------------------------------------


      # -------------- if authentication is oauth ----------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name
that contains client secret. Defaults to 'client_secret', and is automatically added
to request parameters.
      clientSecretParamValue: # [string] Client secret value - Secret value to add
to HTTP requests as the 'client secret' parameter. Stored on disk encrypted, and is
automatically added to request parameters
      authRequestParams: # [array] Extra authentication parameters - OAuth request
parameters added to the POST body. The Content-Type header will automatically be set
to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
```

```
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.

      # ----------------------------------------------------------


      # -------------- if authentication is oauthSecret ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to compute the Authorization header to pass in discover and collect calls. The value ${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name that contains client secret. Defaults to 'client_secret', and is automatically added to request parameters.
      textSecret: # [string] Client secret value (text secret) - Select (or create) a text secret that contains the client secret's value.
      authRequestParams: # [array] Extra authentication parameters - OAuth request parameters added to the POST body. The Content-Type header will automatically be set to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.

      # ----------------------------------------------------------

      timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout, use 0 to disable
      defaultBreakers: # [string] Hidden Default Breakers
      safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.

    # ----------------------------------------------------------


    # -------------- if type is office365_mgmt ---------------

    conf: # [object] [required]
      plan_type: # [string] Subscription plan - Office 365 subscription plan for your organization, typically Enterprise
      tenant_id: # [string] [required] Tenant identifier - Directory ID (tenant identifier) in Azure Active Directory
      app_id: # [string] [required] Application identifier - Identifier of the registered application in Azure Active Directory.
      client_secret: # [string] [required] Client secret - Application key of the registered application.
      publisher_identifier: # [string] Publisher identifier - Optional PublisherIdentifier to use in API requests, defaults to tenant id if not defined.
      content_type: # [string] [required] Content type - The type of content to retrieve from the Office 365 management communications API.

    # ----------------------------------------------------------


    # -------------- if type is office365_service ---------------

    conf: # [object] [required]
      tenant_id: # [string] Tenant identifier - Directory ID (tenant identifier) in Azure Active Directory
      app_id: # [string] [required] Application identifier - Identifier of the registered application in Azure Active Directory.
      client_secret: # [string] [required] Client secret - Application key of the registered application.
      content_type: # [string] [required] Content type - The type of content to retrieve from the Office 365 service communications API.

    # ----------------------------------------------------------


    # -------------- if type is prometheus ---------------

    conf: # [object] [required]
      dimensionList: # [array] Extra Dimensions - Other dimensions to include in events
      username: # [string] Username - Optional username for Basic authentication
      password: # [string] Password - Optional password for Basic authentication
      discoveryType: # [string] Discovery Type - Target discovery mechanism, use static to manually enter a list of targets

      # -------------- if discoveryType is static ---------------

      targetList: # [array] Targets - List of Prometheus targets to pull metrics from, values can be in URL or host[:port] format. For example: http://localhost:9090/metrics, localhost:9090, or localhost. In the cases where just host[:port] are specified, the endpoint will resolve to 'http://host[:port]/metrics'.

      # ----------------------------------------------------------


      # -------------- if discoveryType is dns ---------------

      nameList: # [array] DNS Names - List of DNS names to resolve
      recordType: # [string] Record Type - DNS Record type to resolve
      scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting metrics
      scrapePath: # [string] Metrics Path - Path to use when collecting metrics from discovered targets

      # ----------------------------------------------------------
      # -------------- if discoveryType is ec2 ---------------

      usePublicIp: # [boolean] Use Public IP - Use public IP address for discovered
targets. Set to false if the private IP address should be used.
      scrapeProtocol: # [string] Metrics Protocol - Protocol to use when collecting
metrics
      scrapePort: # [number] Metrics Port - The port number in the metrics URL for
discovered targets.
      scrapePath: # [string] Metrics Path - Path to use when collecting metrics from
discovered targets
      searchFilter: # [array] Search Filter - EC2 Instance Search Filter
        - Name: # [string] Filter Name - Search filter attribute name, see:
https://docs.aws.amazon.com/AWSEC2/latest/APIReference/API_DescribeInstances.html
for more information. Attributes can be manually entered if not present in the drop
down list
          Values: # [array of strings] Filter Values - Search Filter Values
      region: # [string] Region - Region from which to retrieve data.
      awsAuthenticationMethod: # [string] Authentication Method - AWS authentication
method
      enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role credentials
      endpoint: # [string] Endpoint - EC2 service endpoint. If empty, defaults to
AWS' Region-specific endpoint. Otherwise, used to point to a EC2-compatible
endpoint.
      signatureVersion: # [string] Signature version - Signature version to use for
signing EC2 requests

      # --------------------------------------------------------


    # --------------------------------------------------------


    # -------------- if type is rest ---------------

    conf: # [object] [required]
      discovery: # [object]
        discoverType: # [string] Discover type - Defines how task discovery will be
performed. Use None to skip the discovery. Use HTTP Request to make a REST call to
discover tasks. Use Item List to enumerate items for collect to retrieve. Use JSON
Response to manually define discover tasks as a JSON array of objects. Each entry
returned by the discover operation will result in a collect task.

        # -------------- if discoverType is http ---------------

        discoverUrl: # [string] Discover URL - Expression to derive URL to use for
the Discover operation (can be a constant).
        discoverMethod: # [string] Discover method - Discover HTTP method.
        discoverRequestHeaders: # [array] Discover headers - Optional discover
request headers.
          - name: # [string] Name - Header name.
            value: # [string] Value - JavaScript expression to compute the
parameter's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a
constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters
(e.g.,Â earliest) are evaluated as strings.
        discoverDataField: # [string] Discover data field - Path to field in the
response object which contains discover results (e.g.: level1.name), leave blank if
the result is an array.
```

```
        # --------------------------------------------------------


        # -------------- if discoverType is json ---------------

        manualDiscoverResult: # [string] Discover result - Allows hard-coding the
Discover result. Must be a JSON object. Works with the Discover Data field.
        discoverDataField: # [string] Discover data field - Within the response
JSON, name of the field or array element to pull results from. LeaveÂ blank if the
result is an array of values. Sample entry: items, json: { items: [{id: 'first'},
{id: 'second'}] }

        # --------------------------------------------------------


        # -------------- if discoverType is list ---------------

        itemList: # [array of strings] Discover items - Comma-separated list of
items to return from the Discover task. Each item returned will generate a collect
task, and can be referenced using `${id}` in the collect URL, headers, or
parameters.

        # --------------------------------------------------------

      collectUrl: # [string] Collect URL - Expression to derive URL to use for the
collect operation (can be a constant).
      collectMethod: # [string] [required] Collect method - Collect HTTP method.

      # -------------- if collectMethod is get ---------------

      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # --------------------------------------------------------


      # -------------- if collectMethod is post ---------------

      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # --------------------------------------------------------


      # -------------- if collectMethod is post_with_body ---------------

      collectBody: # [string] Collect POST body - Template for POST body to send
with the Collect request. Reference global variables, functions, or parameters from
the Discover response using template params: `${C.vars.myVar}`, or `${Date.now()}`,
`${param}`.
```

```
      # ----------------------------------------------------------


      # -------------- if collectMethod is other ---------------

      collectVerb: # [string] Collect verb - DIY HTTP verb to use for the collect
operation.
      collectBody: # [string] Collect body - Template for body to send with the
Collect request. Reference global variables, functions, or parameters from the
Discover response using template params: `${C.vars.myVar}`, or `${Date.now()}`,
`${param}`
      collectRequestParams: # [array] Collect parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ----------------------------------------------------------

      collectRequestHeaders: # [array] Collect headers - Optional collect request
headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      pagination: # [object]
        type: # [string] Pagination - Select collect pagination scheme

        # -------------- if type is response_body ---------------

        attribute: # [array,string] Response attribute - The name of the attribute
in the response that contains next page information
        maxPages: # [number] Max pages - The maximum number of pages to retrieve,
set to 0 to retrieve all pages

        # ----------------------------------------------------------


        # -------------- if type is response_header_link ---------------

        nextRelationAttribute: # [string] Next page relation name - Relation name
used in the link header that refers to the next page in the result set. In this
example link header, rel="next" to the next page of results:
<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"
        curRelationAttribute: # [string] Current page relation name - Optional
relation name used in the link header that refers to the current result set. In this
example link header, rel="self" refers to the current page of results:
<https://myHost/curPage>; rel="self" <https://myHost/nextPage>; rel="next"
        maxPages: # [number] Max pages - The maximum number of pages to retrieve,
set to 0 to retrieve all pages

        # ----------------------------------------------------------


        # -------------- if type is request_offset ---------------
        offsetField: # [string] Offset field name - Query string parameter that sets
the index from which to begin returning records. E.g.: /api/v1/query?
term=cribl&limit=100&offset=0
        offset: # [number] Starting offset - Offset index from which to start
request. Defaults to undefined, which will start collection from the first record.
        offsetSpacer: # [null]
        limitField: # [string] Limit field name - Query string parameter to set the
number of records retrieved per request. E.g.: /api/v1/query?
term=cribl&limit=100&offset=0
        limit: # [number] Limit - Maximum number of records to collect per request.
        limitSpacer: # [null]
        totalRecordField: # [string] Total record count field name - Identifies the
attribute in the response that contains the total number of records for the query.
        maxPages: # [number] Max pages - The maximum number of pages to retrieve.
Set to 0 to retrieve all pages.
        zeroIndexed: # [boolean] Zero-based index - Toggle to Yes to indicate that
the first record in the requested data is at index 0. The default (No) indicates
index 1.

        # ---------------------------------------------------------



        # -------------- if type is request_page ---------------

        pageField: # [string] Page number field name - Query string parameter that
sets the page index to be returned. E.g.: /api/v1/query?
term=cribl&page_size=100&page_number=0
        page: # [number] Starting page number - Page number from which to start
request. Defaults to undefined, which will start collection from the first page.
        offsetSpacer: # [null]
        sizeField: # [string] Page size field name - Query string parameter to set
the number of records retrieved per request. E.g.: /api/v1/query?
term=cribl&page_size=100&page_number=0
        size: # [number] Page size - Maximum number of records to collect per page.
        limitSpacer: # [null]
        totalPageField: # [string] Total page count field name - The name of the
attribute in the response that contains the total number of pages for the query.
        totalRecordField: # [string] Total record count field name - Identifies the
attribute in the response that contains the total number of records for the query.
        maxPages: # [number] Max pages - The maximum number of pages to retrieve.
Set to 0 to retrieve all pages.
        zeroIndexed: # [boolean] zero-based index - Toggle to Yes to indicate that
the first page in the requested data is at index 0. The default (No) indicates index
1.

        # ---------------------------------------------------------

      authentication: # [string] [required] Authentication - Authentication method
for Discover and Collect REST calls. You can specify API Keyâ¯💳based authentication
by adding the appropriate CollectÂ headers.

      # -------------- if authentication is basic ----------------

      username: # [string] Username - Basic authentication username
      password: # [string] Password - Basic authentication password

      # ---------------------------------------------------------



      # -------------- if authentication is basicSecret ---------------
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your credentials

      # ----------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is
expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ----------------------------------------------------------


      # -------------- if authentication is loginSecret ---------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is
expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored
secret that references your login credentials
      loginBody: # [string] POST body - Template for POST body to send with login
request, ${username} and ${password} are used to specify location of these
attributes in the message
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in
login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to
compute the Authorization header to pass in discover and collect calls. The value
${token} is used to reference the token obtained from login.
      authRequestHeaders: # [array] Authentication headers - Optional authentication
request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.

      # ----------------------------------------------------------


      # -------------- if authentication is oauth ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call
```

```
is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to compute the Authorization header to pass in discover and collect calls. The value ${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name that contains client secret. Defaults to 'client_secret', and is automatically added to request parameters.
      clientSecretParamValue: # [string] Client secret value - Secret value to add to HTTP requests as the 'client secret' parameter. Stored on disk encrypted, and is automatically added to request parameters
      authRequestParams: # [array] Extra authentication parameters - OAuth request parameters added to the POST body. The Content-Type header will automatically be set to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.

      # --------------------------------------------------------


      # -------------- if authentication is oauthSecret ---------------

      loginUrl: # [string] Login URL - URL to use for the OAuth API call. This call is expected to be a POST.
      tokenRespAttribute: # [string] Token attribute - Path to token attribute in login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize expression - JavaScript expression to compute the Authorization header to pass in discover and collect calls. The value ${token} is used to reference the token obtained from login.
      clientSecretParamName: # [string] Client secret parameter - Parameter name that contains client secret. Defaults to 'client_secret', and is automatically added to request parameters.
      textSecret: # [string] Client secret value (text secret) - Select (or create) a text secret that contains the client secret's value.
      authRequestParams: # [array] Extra authentication parameters - OAuth request parameters added to the POST body. The Content-Type header will automatically be set to application/x-www-form-urlencoded.
        - name: # [string] Name - Parameter name.
          value: # [string] Value - JavaScript expression to compute the parameter's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      authRequestHeaders: # [array] Authentication headers - Optional authentication request headers.
        - name: # [string] Name - Header name.
          value: # [string] Value - JavaScript expression to compute the header's value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      # ----------------------------------------------------------

      timeout: # [number] Request timeout (secs) - HTTP request inactivity timeout, use 0 to disable
      useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. Suitable when DNS server returns multiple addresses in sort order.
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector event time filtering when a date range is specified.
      safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.

    # ----------------------------------------------------------


    # -------------- if type is s3 ---------------

    conf: # [object] [required]
      outputName: # [string] Auto-populate from - The name of the predefined Destination that will be used to auto-populate collector settings.
      bucket: # [string] S3 bucket - S3 Bucket from which to collect data.
      region: # [string] Region - Region from which to retrieve data.
      path: # [string] Path - The directory from which to collect data. Templating is supported, e.g.: myDir/${datacenter}/${host}/${app}/. Time-based tokens are also supported, e.g.: myOtherDir/${_time:%Y}/${_time:%m}/${_time:%d}/
      extractors: # [array] Path extractors - Allows using template tokens as context for expressions that enrich discovery results. E.g.: given a template /path/${epoch}, an extractor under key "epoch" with an expression {date: new Date(+value*1000)}, will enrich discovery results with a human readable "date" field
        - key: # [string] Token - A token from the template path, e.g.: epoch
          expression: # [string] Extractor expression - JS expression that receives token under "value" variable, and evaluates to populate event fields, e.g.: {date: new Date(+value*1000)}
      awsAuthenticationMethod: # [string] Authentication Method - AWS authentication method. Choose Auto to use IAM roles.

      # -------------- if awsAuthenticationMethod is manual ---------------

      awsApiKey: # [string] Access key - Access key. If not present, will fall back to env.AWS_ACCESS_KEY_ID, or to the metadata endpoint for IAM creds. Optional when running on AWS.
      awsSecretKey: # [string] Secret key - Secret key. If not present, will fall back to env.AWS_SECRET_ACCESS_KEY, or to the metadata endpoint for IAM creds. Optional when running on AWS.

      # ----------------------------------------------------------


      # -------------- if awsAuthenticationMethod is secret ---------------

      awsSecret: # [string] Secret key pair - Select (or create) a stored secret that references AWS access key and secret key.

      # ----------------------------------------------------------

      endpoint: # [string] Endpoint - S3 service endpoint. If empty, the endpoint will be automatically constructed from the region.
      signatureVersion: # [string] Signature version - Signature version to use for signing S3 requests.
      enableAssumeRole: # [boolean] Enable Assume Role - Use Assume Role
```

```
credentials.
      assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume.
      assumeRoleExternalId: # [string] External ID - External ID to use when
assuming role.
      recurse: # [boolean] Recursive - Whether to recurse through subdirectories.
      maxBatchSize: # [number] Max Batch Size (objects) - Maximum number of metadata
objects to batch before recording as results.
      reuseConnections: # [boolean] Reuse Connections - Whether to reuse connections
between requests, which can improve performance.
      rejectUnauthorized: # [boolean] Reject Unauthorized Certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
      verifyPermissions: # [boolean] Verify bucket permissions - Disable if you can
access files within the bucket but not the bucket itself. Resolves errors of the
form "discover task initialization failed...error: Forbidden".

    # ----------------------------------------------------------


    # -------------- if type is script ---------------

    conf: # [object] [required]
      discoverScript: # [string] Discover Script - Script to discover what to
collect. Should output one task per line in stdout.
      collectScript: # [string] [required] Collect Script - Script to run to perform
data collections. Task passed in as $CRIBL_COLLECT_ARG. Should output results to
stdout.
      shell: # [string] Shell - Shell to use to execute scripts.

    # ----------------------------------------------------------


    # -------------- if type is splunk ---------------

    conf: # [object] [required]
      searchHead: # [string] [required] Search head - Search Head base URL, can be
expression, default is https://localhost:8089.
      search: # [string] Search - Enter Splunk search here. For example:
'index=myAppLogs level=error channel=myApp' OR '| mstats avg(myStat) as myStat WHERE
index=myStatsIndex.'
      earliest: # [string] Earliest - The earliest time boundary for the search. Can
be an exact or relative time. For example: '2022-01-14T12:00:00Z' or '-16m@m'
      latest: # [string] Latest - The latest time boundary for the search. Can be an
exact or relative time. For example: '2022-01-14T12:00:00Z' or '-1m@m'
      endpoint: # [string] [required] Search endpoint - REST API used to create a
search.
      outputMode: # [string] [required] Output mode - Format of the returned output
      collectRequestParams: # [array] Extra parameters - Optional collect request
parameters.
        - name: # [string] Name - Parameter name
          value: # [string] Value - JavaScript expression to compute the parameter's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are
evaluated as strings.
      collectRequestHeaders: # [array] Extra headers - Optional collect request
headers.
        - name: # [string] Name - Header Name
          value: # [string] Value - JavaScript expression to compute the header's
value, normally enclosed in backticks (e.g.,Â `${earliest}`). IfÂ a constant, use
```

```
single quotes (e.g.,Â 'earliest'). ValuesÂ without delimiters (e.g.,Â earliest) are evaluated as strings.
      authentication: # [string] [required] Authentication - Authentication method for Discover and Collect REST calls.

      # -------------- if authentication is basic ---------------

      username: # [string] Username - Basic authentication username
      password: # [string] Password - Basic authentication password

      # ----------------------------------------------------------


      # -------------- if authentication is basicSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a stored secret that references your credentials

      # ----------------------------------------------------------


      # -------------- if authentication is login ---------------

      loginUrl: # [string] Login URL - URL to use for login API call. This call is expected to be a POST.
      username: # [string] Username - Login username
      password: # [string] Password - Login password
      loginBody: # [string] POST Body - Template for POST body to send with login request, ${username} and ${password} are used to specify location of these attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to compute the Authorization header to pass in discover and collect calls. The value ${token} is used to reference the token obtained from login.

      # ----------------------------------------------------------


      # -------------- if authentication is loginSecret ---------------

      loginUrl: # [string] Login URL - URL to use for login API call, this call is expected to be a POST.
      credentialsSecret: # [string] Credentials secret - Select (or create) a stored secret that references your login credentials
      loginBody: # [string] POST Body - Template for POST body to send with login request, ${username} and ${password} are used to specify location of these attributes in the message
      tokenRespAttribute: # [string] Token Attribute - Path to token attribute in login response body. Nested attributes are OK.
      authHeaderExpr: # [string] Authorize Expression - JavaScript expression to compute the Authorization header to pass in discover and collect calls. The value ${token} is used to reference the token obtained from login.

      # ----------------------------------------------------------

      timeout: # [number] Request Timeout (secs) - HTTP request inactivity timeout, use 0 to disable
      useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. Suitable when DNS server returns multiple addresses in sort order.
      disableTimeFilter: # [boolean] Disable time filter - Used to disable collector event time filtering when a date range is specified.

    # ----------------------------------------------------------

    destructive: # [boolean] Destructive - If set to Yes, the collector will delete any files that it collects (where applicable).
  input: # [object]
    type: # [string]
    breakerRulesets: # [array of strings] Event Breaker rulesets - A list of event breaking rulesets that will be applied, in order, to the input data stream.
    staleChannelFlushMs: # [number] Event Breaker buffer timeout - The amount of time (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing the data stream out, as-is, to the Pipelines.
    sendToRoutes: # [boolean] Send to Routes - If set to Yes, events will be sent to normal routing and event processing. Set to No if you want to select a specific Pipeline/Destination combination.

    # -------------- if sendToRoutes is true ---------------

    pipeline: # [string] Pre-Processing Pipeline - Pipeline to process results before sending to routes. Optional.

    # ----------------------------------------------------------


    # -------------- if sendToRoutes is false ---------------

    pipeline: # [string] Pipeline - Pipeline to process results.
    output: # [string] Destination - Destination to send results to.

    # ----------------------------------------------------------

    preprocess: # [object]
      disabled: # [boolean] Disabled - Enable Custom Command
      command: # [string] Command - Command to feed the data through (via stdin) and process its output (stdout)
      args: # [array of strings] Arguments - Arguments
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttle while writing to an output. Also takes values with multiple-byte units, such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value, enclosed in quotes or backticks. (Can evaluate to a constant.)
  type: # [string] Job type - Job type.
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in the Job Inspector.
  removeFields: # [array of strings] Remove Discover fields - List of fields to remove from Discover results. Wildcards (e.g.: aws*) are allowed. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface.
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure condition causes Stream to restart during job execution.
  environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
  schedule: # [object] Schedule - Configuration for a scheduled job.
    enabled: # [boolean] Enabled - Determines whether or not this schedule is enabled.
    # -------------- if enabled is true ---------------

    cronSchedule: # [string] Cron schedule - A cron schedule on which to run this
job.
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of
instances that may be running of this scheduled job at any given time.
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their
next run time, if the system is hitting concurrency limits.
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that
failed with non-fatal errors.
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a task
can be rescheduled.
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run
(e.g., 30, 45s or 15m). Units are seconds, if not specified. Enter 0 for unlimited
time.
      mode: # [string] Mode - Job run mode. Preview will either return up to N
matching results, or will run until capture time T is reached. Discovery will gather
the list of files to turn into streaming tasks, without running the data collection
job. Full Run will run the collection job.

      # -------------- if mode is list ---------------

      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
to routes

      # --------------------------------------------------------


      # -------------- if mode is preview ---------------

      capture: # [object] Capture Settings
        duration: # [number] Capture Time (sec) - Amount of time to keep capture
open, in seconds.
        maxEvents: # [number] Capture Up to N Events - Maximum number of events to
capture.
        level: # [string] Where to capture

      # --------------------------------------------------------

      timeRangeType: # [string] Time range - Time range for scheduled job.
      earliest: # [number,string] Earliest - Earliest time, in local time.
      latest: # [number,string] Latest - Latest time, in local time.
      timestampTimezone: # [string] Range Timezone - Timezone to use for Earliest
and Latest times (defaults to UTC).
      expression: # [string] Filter - A filter for tokens in the provided collect
path and/or the events being collected
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for
small tasks. E.g., bundle five 200KB files into one 1M task.
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for
files above the Lower task bundle size. E.g., bundle five 2MB files into one 10MB
task bundle. Files greater than this size will be assigned to individual tasks.

    # --------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
```

```
Stream.
executor_job: # [object]
  executor: # [object]
    type: # [string] Executor type - The type of executor to run.
    storeTaskResults: # [boolean] Store task results - Determines whether or not to
write task results to disk.
    conf: # [object] Executor-specific settings.
  type: # [string] Job type - Job type.
  ttl: # [string] Time to live - Time to keep the job's artifacts on disk after job
completion. This also affects how long a job is listed in the Job Inspector.
  removeFields: # [array of strings] Remove Discover fields - List of fields to
remove from Discover results. Wildcards (e.g.: aws*) are allowed. This is useful
when discovery returns sensitive fields that should not be exposed in the Jobs user
interface.
  resumeOnBoot: # [boolean] Resume job on boot - Resumes the ad hoc job if a failure
condition causes Stream to restart during job execution.
  environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  schedule: # [object] Schedule - Configuration for a scheduled job.
    enabled: # [boolean] Enabled - Determines whether or not this schedule is
enabled.

    # -------------- if enabled is true ---------------

    cronSchedule: # [string] Cron schedule - A cron schedule on which to run this
job.
    maxConcurrentRuns: # [number] Max concurrent runs - The maximum number of
instances that may be running of this scheduled job at any given time.
    skippable: # [boolean] Skippable - Skippable jobs can be delayed, up to their
next run time, if the system is hitting concurrency limits.
    run: # [object] Run settings

      # -------------- if type is collection ---------------

      rescheduleDroppedTasks: # [boolean] Reschedule tasks - Reschedule tasks that
failed with non-fatal errors.
      maxTaskReschedule: # [number] Max task reschedule - Max number of times a task
can be rescheduled.
      logLevel: # [string] Log Level - Level at which to set task logging.
      jobTimeout: # [string] Job timeout - Maximum time the job is allowed to run
(e.g., 30, 45s or 15m). Units are seconds, if not specified. Enter 0 for unlimited
time.
      mode: # [string] Mode - Job run mode. Preview will either return up to N
matching results, or will run until capture time T is reached. Discovery will gather
the list of files to turn into streaming tasks, without running the data collection
job. Full Run will run the collection job.

      # -------------- if mode is list ---------------

      discoverToRoutes: # [boolean] Send to Routes - If true, send discover results
to routes

      # --------------------------------------------------------



      # -------------- if mode is preview ---------------

      capture: # [object] Capture Settings
        duration: # [number] Capture Time (sec) - Amount of time to keep capture
open, in seconds.
```

```
        maxEvents: # [number] Capture Up to N Events - Maximum number of events to capture.
        level: # [string] Where to capture

      # ---------------------------------------------------------

      timeRangeType: # [string] Time range - Time range for scheduled job.
      earliest: # [number,string] Earliest - Earliest time, in local time.
      latest: # [number,string] Latest - Latest time, in local time.
      timestampTimezone: # [string] Range Timezone - Timezone to use for Earliest and Latest times (defaults to UTC).
      expression: # [string] Filter - A filter for tokens in the provided collect path and/or the events being collected
      minTaskSize: # [string] Lower task bundle size - Limits the bundle size for small tasks. E.g., bundle five 200KB files into one 1M task.
      maxTaskSize: # [string] Upper task bundle size - Limits the bundle size for files above the Lower task bundle size. E.g., bundle five 2MB files into one 10MB task bundle. Files greater than this size will be assigned to individual tasks.

    # ---------------------------------------------------------

  streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
```

> The `workerAffinity` internal parameter defaults to `false`. Cribl automatically sets this to `true` on certain jobs, specifying that collection tasks will be both created and run by the same Worker Node.

;

10.6.8. job-limits.yml

`job-limits.yml` maintains parameters for collection jobs and system tasks. In the UI, you can configure these at global ⚙ **Settings** (lower left) > **General Settings > Job Limits**.

$CRIBL_HOME/default/cribl/job-limits.yml

```
concurrentJobLimit: # [number] Concurrent job limit - The total number of jobs that may run concurrently
concurrentSystemJobLimit: # [number] Concurrent system job limit - The total number of system jobs that may run concurrently
concurrentScheduledJobLimit: # [number] Concurrent scheduled job limit - The total number of scheduled jobs that may run concurrently. Limit is relative to concurrent job limit.
concurrentTaskLimit: # [number] Concurrent task limit - The total number of tasks that a worker process may run concurrently
concurrentSystemTaskLimit: # [number] Concurrent system task limit - The number of system tasks that a worker process may run concurrently
maxTaskPerc: # [number] Max task usage percentage - Value from 0 to 1 representing the percentage of total tasks within the system a job may consume
taskPollTimeoutMs: # [number] Task poll timeout - The number of milliseconds the worker's task handler will wait to receive a task before retrying the request for a task.
jobArtifactsReaperPeriod: # [string] Artifact reaper period - Time period at which the system attempts to reap stale disk artifacts belonging to the jobs
finishedJobArtifactsLimit: # [number] Finished job artifacts limit - Maximum number of finished job artifacts to keep on disk.
finishedTaskArtifactsLimit: # [number] Finished task artifacts limit - Maximum number of finished task artifacts to keep on disk per job on each worker node.
taskManifestFlushPeriodMs: # [number] Manifest flush period - The rate at which a job's task manifest should be refreshed in milliseconds.
taskManifestMaxBufferSize: # [number] Manifest max buffer size - The maximum number of tasks the task manifest can hold in memory before flushing to disk.
taskManifestReadBufferSize: # [string] Manifest reader buffer size - The number of bytes the task manifest reader should pull from disk.
schedulingPolicy: # [string] Job dispatching - The method by which tasks are assigned to worker processes to complete the work
jobTimeout: # [string] Job timeout - Maximum time a job is allowed to run. Defaults to 0, for unlimited time. Units are seconds if not specified. Sample entries: 30, 45s, 15m.
taskHeartbeatPeriod: # [number] Task heartbeat period - The heartbeat period (in seconds) for tasks to report back to the leader/API.
```

;

10.6.9. licenses.yml

`licenses.yml` maintains a list of Cribl Stream licenses.

$CRIBL_HOME/default/cribl/licenses.yml

```
licenses: # [array of string] - list of licenses
```

;

10.6.10. 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 at global ⚙ **Settings** (lower left) > **General Settings > Limits**.

$CRIBL_HOME/default/cribl/limits.yml

```
samples: # [object] Samples
  maxSize: # [string] Max sample size - Maximum file size for the sample, in binary units (KB, MB). (Max. 3MB.)
minFreeSpace: # [string] Min free disk space - The minimum amount of disk space in the system before various features will take measures to prevent disk usage (KB, MB, etc.).
metricsGCPeriod: # [string] Metrics GC period - The interval on which the system attempts to free memory, by pruning stale metrics from the Stream system metrics store.
metricsMaxCardinality: # [number] Metrics cardinality limit - The system's allowed number of permutations of a given metric name.
metricsMaxDiskSpace: # [string] Metrics max disk space - Maximum allowed disk space for persisting metrics to disk.
metricsWorkerIdBlacklist: # [array of strings] Metrics worker tracking - List of metric names for which to disable tracking of Worker Node ID. Supports wildcards.
metricsNeverDropList: # [array of strings] Metrics never drop list - List metric names for which to ensure delivery. Supports wildcards.
metricsFieldsBlacklist: # [array of strings] Disable field metrics - List of event fields for which to disable metric collection.
metricsDirectory: # [string] Metrics directory - Directory to store metrics on disk.
cpuProfileTTL: # [string] CPU profile TTL - The time-to-live for collected CPU profiles.
eventsMetadataSources: # [array of strings] Event metadata sources - List of event metadata sources to enable.
```

;

10.6.11. logger.yml

`logger.yml` maintains logging levels and redactions, per channel. In the UI, you can configure these at global ⚙ **Settings** (lower left) > **System > Logging**.

$CRIBL_HOME/default/cribl/logger.yml

```
redactFields: # [array of strings] - list of fields to redact
redactLabel: # [string] - redact label
channels: # [object] Logger channels as logger ID/log level pairs. Log levels: error, warn, info, http, verbose, debug, silly
  DEFAULT: info
  input:DistMaster: debug
  output:DistWorker: debug
```

;

10.6.12. 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]
```

;

10.6.13. 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]
```

;

10.6.14. 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
will be used whenever a nonexistent/deleted output is referenced.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  webhook_output: # [object]
    type: # [string] Output Type
    url: # [string] URL - URL to send events to.
    method: # [string] Method - The method to use when sending events. Defaults to
POST.
    format: # [string] Format - Specifies how to format events before sending out.
Defaults to NDJSON.

    # -------------- if format is custom ---------------

    customSourceExpression: # [string] Source expression - Expression to evaluate as
event. E.g., `${fieldA}, ${fieldB}`. Defaults to __httpOut (i.e. value of field
__httpOut).
    customDropWhenNull: # [boolean] Drop when null - Whether or not to drop events
when the source expression evaluates to null.
    customEventDelimiter: # [string] Event delimiter - Delimiter string to insert
between individual events. Defaults to newline character.
    customContentType: # [string] Content type - Content type to use for request.
Defaults to application/x-ndjson. Any content types set in Advanced Settings > Extra
HTTP headers will override this entry.

    # ---------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
```

```
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    authType: # [string] Authentication type - The authentication method to use for
the HTTP request. Defaults to None.

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ---------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # ---------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # ---------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ---------------------------------------------------------
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  devnull_output: # [object]
    type: # [string] Output Type
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  syslog_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Protocol - The network protocol to use for sending out
syslog messages

    # -------------- if protocol is tcp ---------------

    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.
```

```
    # ----------------------------------------------------------


    # -------------- if protocol is udp ---------------

    host: # [string] Address - The hostname of the receiver
    port: # [number] Port - The port to connect to on the provided host
    maxRecordSize: # [number] Max Record Size - Maximum size of syslog messages. If
max record size is > than MTU then UDP packets can be fragmented across, set this
value  <= MTU to avoid fragmentation.

    # ----------------------------------------------------------

    facility: # [number] Facility - Default value for message facility, will be
overwritten by value of __facility if set. Defaults to user.
    severity: # [number] Severity - Default value for message severity, will be
overwritten by value of __severity if set. Defaults to notice.
    appName: # [string] App Name - Default value for application name, will be
overwritten by value of __appname if set. Defaults to Cribl.
    messageFormat: # [string] Message Format - The syslog message format depending
on the receiver's support
    timestampFormat: # [string] Timestamp Format - Timestamp format to use when
serializing event's time field
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    octetCountFraming: # [boolean] Octet count framing - When enabled, messages will
be prefixed with the byte count of the message. Otherwise, no prefix will be set,
and the message will be appended with a \n.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_output: # [object]
    type: # [string] Output Type
    host: # [string] Address - The hostname of the receiver
    port: # [number] [required] Port - The port to connect to on the provided host
    nestedFields: # [string] Nested field serialization - Specifies how to serialize
nested fields into index-time fields.
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
```

```
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in
multiple-metric format in a single event. Supported in Splunk 8.0 and above.
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is
shutting down and stop sending data. This helps minimize data loss during shutdown.
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
advertise during handshake.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Shared secret token to use when establishing
a connection to a Splunk indexer.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------
    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  splunk_lb_output: # [object]
    type: # [string] Output Type
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.
    nestedFields: # [string] Nested field serialization - Specifies how to serialize
nested fields into index-time fields.
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    enableMultiMetrics: # [boolean] Output multiple metrics - Output metrics in
multiple-metric format in a single event. Supported in Splunk 8.0 and above.
    enableACK: # [boolean] Minimize in-flight data loss - Check if indexer is
```

```
shutting down and stop sending data. This helps minimize data loss during shutdown.
    maxS2Sversion: # [string] Max S2S version - The highest S2S protocol version to
advertise during handshake.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    indexerDiscovery: # [boolean] Indexer Discovery - Automatically discover
indexers in indexer clustering environment.

    # -------------- if indexerDiscovery is true ---------------

    indexerDiscoveryConfigs: # [object]  - List of configurations to set up indexer
discovery in Splunk Indexer clustering environment.
      site: # [string] [required] Site - Clustering site of the indexers from where
indexers need to be discovered. In case of single site cluster, it defaults to
'default' site.
      masterUri: # [string] Cluster Manager URI - Full URI of Splunk cluster Manager
(scheme://host:port). E.g.: https://managerAddress:8089
      refreshIntervalSec: # [number] [required] Refresh Period - Time interval in
seconds between two consecutive indexer list fetches from cluster Manager.
      authType: # [string] Authentication method - Enter a token directly, or
provide a secret referencing a token
      authToken: # [string] Auth token - Authentication token required to
authenticate to cluster Manager for indexer discovery.

      # -------------- if authType is manual ---------------


      # ----------------------------------------------------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

      # -------------- if authType is secret ---------------


      # ----------------------------------------------------------


    # ----------------------------------------------------------
    # -------------- if indexerDiscovery is false ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the current host from the list of any resolved hostnames.
    hosts: # [array] Destinations - Set of Splunk indexers to load-balance data to.
      - host: # [string] Address - The hostname of the receiver.
        port: # [number] Port - The port to connect to on the provided host.
        tls: # [string] TLS - Whether to inherit TLS configs from group setting or disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a TLS connection. If not specified, defaults to connection host (iff not an IP); otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent connections (per worker process). A random set of IPs will be picked on every DNS resolution period. Use 0 for unlimited.

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide a secret referencing a token
    authToken: # [string] Auth token - Shared secret token to use when establishing a connection to a Splunk indexer.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  splunk_hec_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    url: # [string] Splunk HEC Endpoint - URL to a Splunk HEC endpoint to send events to, e.g., http://localhost:8088/services/collector/event
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.

    # ----------------------------------------------------------



    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the current host from the list of any resolved hostnames.
    urls: # [array] Splunk HEC Endpoints
      - url: # [string] HEC Endpoint - URL to a Splunk HEC endpoint to send events to, e.g., http://localhost:8088/services/collector/event
        weight: # [number] Load Weight - The weight to use for load-balancing purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How far back in time to keep traffic stats for load balancing purposes.

    # ----------------------------------------------------------

    nextQueue: # [string] Next Processing Queue - Which Splunk processing queue to send the events after HEC processing
    tcpRouting: # [string] Default _TCP_ROUTING - Set the value of _TCP_ROUTING for events that does not have _ctrl._TCP_ROUTING set
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    enableMultiMetrics: # [boolean] Output multi-metrics - Output metrics in multiple-metric format, supported in Splunk 8.0 and above to allow multiple metrics in a single event.
    authType: # [string] Authentication method - Enter a token directly, or provide a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] HEC Auth token - Splunk HEC authentication token
    # ----------------------------------------------------------


    # -------------- if authType is secret ---------------

    textSecret: # [string] HEC Auth token (text secret) - Select (or create) a
stored text secret

    # ----------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  tcpjson_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    host: # [string] Address - The hostname of the receiver
    port: # [number] Port - The port to connect to on the provided host

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    hosts: # [array] Destinations - Set of hosts to load-balance data to.
      - host: # [string] Address - The hostname of the receiver.
```

```
        port: # [number] Port - The port to connect to on the provided host.
        tls: # [string] TLS - Whether to inherit TLS configs from group setting or disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a TLS connection. If not specified, defaults to connection host (iff not an IP); otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent connections (per worker process). A random set of IPs will be picked on every DNS resolution period. Use 0 for unlimited.

    # ----------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttle while writing to an output. Also takes values with multiple-byte units, such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when connecting

      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds) to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes before the internally generated authentication token expires, valid values between 1 and 60
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Optional authentication token to include as
part of the connection header

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  wavefront_output: # [object]
    type: # [string] Output Type
    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] Auth token - WaveFront API authentication token (see [here]
(https://docs.wavefront.com/wavefront_api.html#generating-an-api-token))

    # ----------------------------------------------------------


    # -------------- if authType is secret ---------------
    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

    # ----------------------------------------------------------

    domain: # [string] Domain name - WaveFront domain name, e.g. "longboard"
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  signalfx_output: # [object]
    type: # [string] Output Type
    authType: # [string] Authentication method - Enter a token directly, or provide a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] Auth token - SignalFx API access token (see [here](https://docs.signalfx.com/en/latest/admin-guide/tokens.html#working-with-access-tokens))

    # ---------------------------------------------------------


    # -------------- if authType is secret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

    # ---------------------------------------------------------

    realm: # [string] Realm - SignalFx realm name, e.g. "us0"
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  filesystem_output: # [object]
    type: # [string] Output Type
    destPath: # [string] Output location - Final destination for the output files
    stagePath: # [string] Staging location - Filesystem location in which to buffer
files before compressing and moving to final destination. Use performant stable
storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ----------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â¯ if present â¯ otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ----------------------------------------------------------


    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
```

```
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments. E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher values improve compression. Imposes a target, not a strict limit; pages final size may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows that were skipped due to data format mismatch. Must have Logging set to Debug to see output.

    # ----------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression to define the output filename suffix (can be constant).  The `__format` variable refers to the value of the `Data format` field (`json` or `raw`).  The `__compression` field refers to the kind of compression being used (`none` or `gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time to write to a file. Files open for longer than this will be closed and moved to final output location.
    maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time to keep inactive files open. Files open for longer than this will be closed and moved to final output location.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open concurrently. When over, the oldest open files will be closed and moved to final output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  s3_output: # [object]
    type: # [string] Output Type
    bucket: # [string] S3 bucket name - Name of the destination S3 bucket. Must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at init time. E.g., referencing a Global Variable: `myBucket-${C.vars.myVar}`.
    region: # [string] Region - Region where the S3 bucket is located.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key
    # --------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # --------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    endpoint: # [string] Endpoint - S3 service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing S3 requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for S3 - Use Assume Role credentials to
access S3
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # --------------------------------------------------------

    destPath: # [string] [required] Key prefix - Prefix to append to files before
uploading. Must be a JavaScript expression (which can evaluate to a constant value),
enclosed in quotes or backticks. Can be evaluated only at init time. E.g.,
referencing a Global Variable: `myKeyPrefix-${C.vars.myVar}`.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    serverSideEncryption: # [string] Server side encryption - Server-side encryption
for uploaded objects.

    # -------------- if serverSideEncryption is aws:kms ---------------

    kmsKeyId: # [string] KMS key ID - ID or ARN of the KMS customer managed key to
use for encryption

    # --------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
```

```
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â¯🪪 if present â¯🪪 otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ----------------------------------------------------------



    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ----------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
to write to a file. Files open for longer than this will be closed and moved to
final output location.
    maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
to keep inactive files open. Files open for longer than this will be closed and
moved to final output location.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
```

```
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_blob_output: # [object]
    type: # [string] Output Type
    containerName: # [string] Container name - A container organizes a set of blobs,
similar to a directory in a file system.
    createContainer: # [boolean] Create container - Creates the configured container
in Azure Blob Storage if it does not already exist.
    destPath: # [string] [required] Blob prefix - Root directory prepended to path
before uploading.
    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ---------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â¯🧾 if present â¯🧾 otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ---------------------------------------------------------



    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.
```

```
    # ----------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
to write to a file. Files open for longer than this will be closed and moved to
final output location.
    maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
to keep inactive files open. Files open for longer than this will be closed and
moved to final output location.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    authType: # [string] Authentication method - Enter connection string directly,
or select a stored secret
    connectionString: # [string] Connection string - Enter your Azure Storage
account connection string. If left blank, Stream will fall back to
env.AZURE_STORAGE_CONNECTION_STRING.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Connection string (text secret) - Select (or create) a
stored text secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_logs_output: # [object]
    type: # [string] Output Type
    logType: # [string] Log Type - The Log Type of events sent to this LogAnalytics
workspace. Can be overwritten by event field __logType.
    resourceId: # [string] Resource ID - Optional Resource ID of the Azure resource
the data should be associated with, can be overridden by event field __resourceId.
This populates the _ResourceId property and allows the data to be included in
resource-centric queries. If this field isn't specified, the data will not be
included in resource-centric queries.
```

```
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    authType: # [string] Authentication method - Enter workspace ID and workspace key directly, or select a stored secret
    workspaceId: # [string] Workspace ID - Azure Log Analytics Workspace ID. See Azure Dashboard WorkspaceÂ > Advanced settings.
    workspaceKey: # [string] Workspace key - Azure Log Analytics Workspace Primary or Secondary Shared Key. See Azure Dashboard WorkspaceÂ > Advanced settings.

    # -------------- if authType is manual ---------------


    # ---------------------------------------------------------
    keypairSecret: # [string] Secret key pair - Select (or create) a stored secret that references your access key and secret key.

    # -------------- if authType is secret ---------------


    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  kinesis_output: # [object]
    type: # [string] Output Type
    streamName: # [string] Stream Name - Kinesis stream name to send events to.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # --------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that references your access key and secret key.

    # --------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the Kinesis stream is located
    endpoint: # [string] Endpoint - Kinesis stream service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to Kinesis stream-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signing Kinesis stream requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to reject certificates that cannot be verified against a valid CA (e.g., self-signed certificates).
    enableAssumeRole: # [boolean] Enable for Kinesis stream - Use Assume Role credentials to access Kinesis stream
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming role
    concurrency: # [number] Put request concurrency - Maximum number of ongoing put requests before blocking.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
```

```
(KB) of each individual record before compression. For non-compressible data 1MB is
the max recommended size
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # -----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  honeycomb_output: # [object]
    type: # [string] Output Type
    dataset: # [string] Dataset name - Name of the dataset to send events to â¯💳
e.g., observability
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
```

```
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    team: # [string] API key - Team API key where the dataset belongs

    # -------------- if authType is manual ---------------


    # ---------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  azure_eventhub_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Event Hubs Kafka brokers to
connect to, eg. yourdomain.servicebus.windows.net:9093. The hostname can be found in
the host portion of the primary or secondary connection string in Shared Access
```

```
Policies.
    topic: # [string] [required] Event Hub Name - The name of the Event Hub (a.k.a.
Kafka Topic) to publish events. Can be overwritten using field __topicOut.
    ack: # [number] Acknowledgments - Control the number of required acknowledgments
    format: # [string] Record data format - Format to use to serialize events before
writing to the Event Hubs Kafka brokers.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Event Hubs brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable authentication.

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.
PLAIN is the only mechanism currently supported for Event Hubs Kafka brokers.
      username: # [string] Username - The username for authentication. For Event
Hubs, this should always be $ConnectionString.
      authType: # [string] Authentication method - Enter password directly, or
select a stored secret

      # -----------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - For Event Hubs, this
should always be false.

      # -----------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  google_chronicle_output: # [object]
    type: # [string] Output Type
    authenticationMethod: # [string] Authentication Method

    # -------------- if authenticationMethod is manual ---------------

    apiKey: # [string] API key - Organization's API key in Google Chronicle

    # ----------------------------------------------------------


    # -------------- if authenticationMethod is secret ---------------

    apiKeySecret: # [string] API key (text secret) - Select (or create) a stored
text secret

    # ----------------------------------------------------------

    logType: # [string] Log type - Log type value to send to Chronicle. Can be
overwritten by event field __logType.
    logTextField: # [string] Log text field - Name of the event field that contains
the log text to send. If not specified, Stream sends a JSON representation of the
whole event.
    logFormatType: # [string] Send events as
    region: # [string] Region - Regional endpoint to send events to
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
```

```
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  google_cloud_storage_output: # [object]
    type: # [string] Output Type
    bucket: # [string] Bucket name - Name of the destination Bucket. This value can
be a constant or a JavaScript expression that can only be evaluated at init time.
E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
    region: # [string] [required] Region - Region where the bucket is located.
    endpoint: # [string] [required] Endpoint - Google Cloud Storage service
endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing Google Cloud Storage requests.
    awsAuthenticationMethod: # [string] Authentication method

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - HMAC access Key
    awsSecretKey: # [string] Secret - HMAC secret

    # ---------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
```

```
references your access key and secret key.

    # ----------------------------------------------------------

    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    destPath: # [string] [required] Key prefix - Prefix to append to files before
uploading. Must be a JavaScript expression (which can evaluate to a constant value),
enclosed in quotes or backticks. Can be evaluated only at init time. E.g.,
referencing a Global Variable: `myKeyPrefix-${C.vars.myVar}`.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ----------------------------------------------------------

    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â¯💳 if present â¯💳 otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ----------------------------------------------------------



    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
```

```
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ----------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
to write to a file. Files open for longer than this will be closed and moved to
final output location.
    maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
to keep inactive files open. Files open for longer than this will be closed and
moved to final output location.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  google_pubsub_output: # [object]
    type: # [string] Output Type
    topicName: # [string] Topic ID - ID of the topic to send events to.
    createTopic: # [boolean] Create topic - If enabled, create topic if it does not
exist.
    orderedDelivery: # [boolean] Ordered delivery - If enabled, send events in the
order they were added to the queue. For this to work correctly, the process
receiving events must have ordering enabled.
    region: # [string] Region - Region to publish messages to. Select 'default' to
allow Google to auto-select the nearest region. When using ordered delivery, the
selected region must be allowed by message storage policy.
    googleAuthMethod: # [string] Authentication Method - Google authentication
method. Choose Auto to use environment variables PUBSUB_PROJECT and
PUBSUB_CREDENTIALS.

    # -------------- if googleAuthMethod is manual ---------------

    serviceAccountCredentials: # [string] Service account credentials - Contents of
service account credentials (JSON keys) file downloaded from Google Cloud. To upload
a file, click the upload button at this field's upper right. As an alternative, you
can use environment variables (see [here](https://googleapis.dev/ruby/google-cloud-
pubsub/latest/file.AUTHENTICATION.html)).

    # ----------------------------------------------------------
    # -------------- if googleAuthMethod is secret ---------------

    secret: # [string] Service account credentials (text secret) - Select (or create) a stored text secret

    # --------------------------------------------------------

    batchSize: # [number] Batch size - The maximum number of items the Google API should batch before it sends them to the topic.
    batchTimeout: # [number] Batch timeout (ms) - The maximum amount of time, in milliseconds, that the Google API should wait to send a batch (if the Batch size is not reached).
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches before blocking.
    maxRecordSizeKB: # [number] Max batch size (KB) - Maximum size (KB) of batches to send.
    maxInProgress: # [number] Max concurrent requests - The maximum number of in-progress API requests before backpressure is applied.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  kafka_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Kafka brokers to connect to, e.g., kafkaBrokerHost:9092.
    topic: # [string] [required] Topic - The topic to publish events to. Can be overridden using the __topicOut field.
    ack: # [number] Acknowledgments - Control the number of required acknowledgments.
    format: # [string] Record data format - Format to use to serialize events before
```

```
writing to Kafka.
    compression: # [string] Compression - Codec to use to compress the data before
sending to Kafka.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Kafka brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
      disabled: # [boolean] Disabled - Enable Schema Registry

      # -------------- if disabled is false ----------------

      schemaRegistryURL: # [string] Schema Registry URL - URL for access to the
Confluent Schema Registry, i.e.: http://localhost:8081
      defaultKeySchemaId: # [number] Default key schema ID - Used when
__keySchemaIdOut is not present, to transform key values, leave blank if key
transformation is not required by default.
      defaultValueSchemaId: # [number] Default value schema ID - Used when
__valueSchemaIdOut is not present, to transform _raw, leave blank if value
transformation is not required by default.
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ----------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that
are not authorized by a CA in the CA certificate path, or by another trusted CA
(e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined
certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in
which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private
key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

        # --------------------------------------------------------


      # --------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for
a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a
successful request.
    sasl: # [object] Authentication - Authentication parameters to use when
connecting to brokers. Using TLS is highly recommended.
```

```
      disabled: # [boolean] Disabled - Enable Authentication

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

      # ---------------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when connecting

      # ---------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
```

```
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  confluent_cloud_output: # [object]
    type: # [string] Output Type
    brokers: # [array of strings] Brokers - List of Confluent Cloud brokers to
connect to, e.g., yourAccount.confluent.cloud:9092.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ---------------------------------------------------------

    topic: # [string] [required] Topic - The topic to publish events to. Can be
overridden using the __topicOut field.
    ack: # [number] Acknowledgments - Control the number of required
acknowledgments.
    format: # [string] Record data format - Format to use to serialize events before
writing to Kafka.
    compression: # [string] Compression - Codec to use to compress the data before
sending to Kafka.
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size
(KB) of each record batch before compression. Setting should be < message.max.bytes
settings in Kafka brokers.
    flushEventCount: # [number] Max events per batch - Maximum number of events in a
batch before forcing a flush.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    kafkaSchemaRegistry: # [object] Kafka Schema Registry Authentication
      disabled: # [boolean] Disabled - Enable Schema Registry

      # -------------- if disabled is false ---------------

      schemaRegistryURL: # [string] Schema Registry URL - URL for access to the
Confluent Schema Registry, i.e.: http://localhost:8081
      defaultKeySchemaId: # [number] Default key schema ID - Used when
__keySchemaIdOut is not present, to transform key values, leave blank if key
```

```
transformation is not required by default.
      defaultValueSchemaId: # [number] Default value schema ID - Used when __valueSchemaIdOut is not present, to transform _raw, leave blank if value transformation is not required by default.
      tls: # [object] TLS settings (client side)
        disabled: # [boolean] Disabled

        # -------------- if disabled is false ---------------

        rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
        servername: # [string] Server name (SNI) - Server name for the SNI (Server Name Indication) TLS extension. It must be a host name, and not an IP address.
        certificateName: # [string] Certificate name - The name of the predefined certificate.
        caPath: # [string] CA certificate path - Path on client in which to find CA certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
        privKeyPath: # [string] Private key path (mutual auth) - Path on client in which to find the private key to use. PEM format. Can reference $ENV_VARS.
        certPath: # [string] Certificate path (mutual auth) - Path on client in which to find certificates to use. PEM format. Can reference $ENV_VARS.
        passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
        minVersion: # [string] Minimum TLS version - Minimum TLS version to use when connecting
        maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when connecting

        # --------------------------------------------------------


      # --------------------------------------------------------

    connectionTimeout: # [number] Connection timeout (ms) - Maximum time to wait for a successful connection.
    requestTimeout: # [number] Request timeout (ms) - Maximum time to wait for a successful request.
    sasl: # [object] Authentication - Authentication parameters to use when connecting to brokers. Using TLS is highly recommended.
      disabled: # [boolean] Disabled - Enable Authentication

      # -------------- if disabled is false ---------------

      mechanism: # [string] SASL mechanism - SASL authentication mechanism to use.

      # --------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  elastic_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    url: # [string] Bulk API URL or Cloud ID - Enter Cloud ID or URL to an Elastic
cluster to send events to â¯ e.g., http://elastic:9200/_bulk
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Bulk API URLs
      - url: # [string] URL - URL to an Elastic node to send events to â¯ e.g.,
http://elastic:9200/_bulk
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # ----------------------------------------------------------

    index: # [string] Index or Data Stream - Index or Data Stream to send events to.
Must be a JavaScript expression (which can evaluate to a constant value), enclosed
in quotes or backticks. Can be overwritten by an event's __index field.
    docType: # [string] Type - Document type to use for events. Can be overwritten
by an event's __type field
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
```

```
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    extraParams: # [array] Extra Parameters - Extra Parameters.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    auth: # [object]
      disabled: # [boolean] Authentication Disabled

      # -------------- if disabled is false ---------------

      authType: # [string] Authentication method - Enter credentials directly, or
select a stored secret

      # ---------------------------------------------------------

    elasticVersion: # [string] Elastic Version - Optional Elasticsearch version,
used to format events. If not specified, will auto-discover version.
    elasticPipeline: # [string] Elastic pipeline - Optional Elasticsearch
destination pipeline
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
```

```
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  newrelic_output: # [object]
    type: # [string] Output Type
    region: # [string] Region - Which New Relic region endpoint to use.
    logType: # [string] Log type - Name of the logtype to send with events, e.g.:
observability, access_log. The event's 'sourcetype' field (if set) will override
this value.
    messageField: # [string] Log message field - Name of field to send as log
message value. If not present, event will be serialized and sent as JSON.
    metadata: # [array] Fields - Fields to add to events from this input.
      - name: # [string] Name - Field name
        value: # [string] Value - JavaScript expression to compute field's value,
enclosed in quotes or backticks. (Can evaluate to a constant.)
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
```

```
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - New Relic API key. Can be overridden using
__newRelic_apiKey field.

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  newrelic_events_output: # [object]
    type: # [string] Output Type
    region: # [string] [required] Region - Which New Relic region endpoint to use.
    accountId: # [string] Account ID - New Relic account ID
    eventType: # [string] [required] Event type - Default eventType to use when not
present in an event. For more information, see [here]
(https://docs.newrelic.com/docs/telemetry-data-platform/custom-data/custom-
events/data-requirements-limits-custom-event-data/#reserved-words).
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
```

```
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - New Relic API key. Can be overridden using
__newRelic_apiKey field.

    # -------------- if authType is manual ---------------


    # ---------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  influxdb_output: # [object]
```

```
    type: # [string] Output Type
    url: # [string] Write API URL - URL of an InfluxDB cluster to send events to, e.g., http://localhost:8086/write
    useV2API: # [boolean] Use v2 API - The v2 API can be enabled with InfluxDB versions 1.8 and later.

    # -------------- if useV2API is false ---------------

    database: # [string] Database - Database to write to.

    # --------------------------------------------------------


    # -------------- if useV2API is true ---------------

    bucket: # [string] Bucket - Bucket to write to.
    org: # [string] Organization - Organization ID for this bucket.

    # --------------------------------------------------------

    timestampPrecision: # [string] Timestamp precision - Sets the precision for the supplied Unix time values. Defaults to milliseconds.
    dynamicValueFieldName: # [boolean] Dynamic value fields - Enabling this will pull the value field from the metric name. E,g, 'db.query.user' will use 'db.query' as the measurement and 'user' as the value field.
    valueFieldName: # [string] Value field name - Name of the field in which to store the metric when sending to InfluxDB. If dynamic generation is enabled and fails, this will be used as a fallback.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication type - InfluxDB authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ----------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # ----------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret that references your credentials

    # ----------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text secret

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  cloudwatch_output: # [object]
    type: # [string] Output Type
    logGroupName: # [string] Log group name - CloudWatch log group to associate events with
    logStreamName: # [string] [required] Log stream prefix - Prefix for CloudWatch log stream name. This prefix will be used to generate a unique log stream name per cribl instance, for example: myStream_myHost_myOutputId
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # --------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that references your access key and secret key.

    # --------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] [required] Region - Region where the CloudWatchLogs is located
    endpoint: # [string] Endpoint - CloudWatchLogs service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to CloudWatchLogs-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for signing CloudWatchLogs requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to reject certificates that cannot be verified against a valid CA (e.g., self-signed certificates).
    enableAssumeRole: # [boolean] Enable for CloudWatchLogs - Use Assume Role credentials to access CloudWatchLogs
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming role
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches before blocking
    maxRecordSizeKB: # [number] Max record size (KB, uncompressed) - Maximum size (KB) of each individual record before compression. For non compressible data 1MB is the max recommended size
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max record size.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
```

```
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  minio_output: # [object]
    type: # [string] Output Type
    endpoint: # [string] [required] MinIO endpoint - MinIO service url (e.g.
http://minioHost:9000)
    bucket: # [string] MinIO bucket name - Name of the destination MinIO bucket.
This value can be a constant or a JavaScript expression that can only be evaluated
at init time. E.g. referencing a Global Variable: `myBucket-${C.vars.myVar}`.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication
method

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # ----------------------------------------------------------



    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that
references your access key and secret key.

    # ----------------------------------------------------------

    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - Region where the MinIO service/cluster is located
    stagePath: # [string] [required] Staging location - Filesystem location in which
to buffer files, before compressing and moving to final destination. Use performant
stable storage.
    addIdToStagePath: # [boolean] Add output ID - Append output's ID to staging
location.
    removeEmptyDirs: # [boolean] Remove staging dirs - Remove empty staging
directories after moving files.

    # -------------- if removeEmptyDirs is true ---------------

    emptyDirCleanupSec: # [number] Staging cleanup period - How often (secs) to
clean-up empty directories when 'Remove Staging Dirs' is enabled.

    # ----------------------------------------------------------
    destPath: # [string] [required] Key prefix - Root directory to prepend to path
before uploading. Enter a constant, or a JS expression enclosed in quotes or
backticks.
    signatureVersion: # [string] Signature version - Signature version to use for
signing MinIO requests.
    objectACL: # [string] Object ACL - Object ACL to assign to uploaded objects.
    storageClass: # [string] Storage class - Storage class to select for uploaded
objects.
    serverSideEncryption: # [string] Server-side encryption - Server-side encryption
for uploaded objects.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    partitionExpr: # [string] Partitioning expression - JS expression defining how
files are partitioned and organized. Default is date-based. If blank, Stream will
fall back to the event's __partition field value â¯💳 if present â¯💳 otherwise to
each location's root directory.
    format: # [string] Data format - Format of the output data.

    # -------------- if format is json ---------------

    compress: # [string] Compress - Choose data compression format to apply before
moving files to final destination.

    # ----------------------------------------------------------



    # -------------- if format is parquet ---------------

    parquetSchema: # [string] Parquet schema - Select a Parquet schema. New schemas
can be uploaded under Processing > Knowledge > Parquet Schemas
    parquetRowGroupSize: # [string] Row group size - Ideal memory size for row group
segments. E.g., 128MB or 1GB. Affects memory use when writing. Imposes a target, not
a strict limit; row groups final size may be larger or smaller.
    parquetPageSize: # [string] Page size - Ideal memory size for page segments.
E.g., 1MB or 128MB. Generally, lower values improve reading speed, while higher
values improve compression. Imposes a target, not a strict limit; pages final size
may be larger or smaller.
    spacer: # [null]
    parquetVersion: # [string] Parquet version - Determines which data types are
supported and how they are represented.
    parquetDataPageVersion: # [string] Data page version - Serialization format of
data pages. Note that not all reader implemtations support Data page V2.
    shouldLogInvalidRows: # [boolean] Log invalid rows - Output up to 20 unique rows
that were skipped due to data format mismatch. Must have Logging set to Debug to see
output.

    # ----------------------------------------------------------

    baseFileName: # [string] File name prefix expression - JavaScript expression to
define the output filename prefix (can be constant).
    fileNameSuffix: # [string] File name suffix expression - JavaScript expression
to define the output filename suffix (can be constant).  The `__format` variable
refers to the value of the `Data format` field (`json` or `raw`).  The
`__compression` field refers to the kind of compression being used (`none` or
`gzip`)
    maxFileSizeMB: # [number] Max file size (MB) - Maximum uncompressed output file
```

```
size. Files of this size will be closed and moved to final output location.
    maxFileOpenTimeSec: # [number] Max file open time (Sec) - Maximum amount of time
to write to a file. Files open for longer than this will be closed and moved to
final output location.
    maxFileIdleTimeSec: # [number] Max file idle time (Sec) - Maximum amount of time
to keep inactive files open. Files open for longer than this will be closed and
moved to final output location.
    maxOpenFiles: # [number] Max open files - Maximum number of files to keep open
concurrently. When over, the oldest open files will be closed and moved to final
output location.
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  statsd_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------------------------------------------------

    host: # [string] [required] Host - The hostname of the destination.
    port: # [number] [required] Port - Destination port.
    mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  statsd_ext_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
```

```
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # ----------------------------------------------------------

    host: # [string] [required] Host - The hostname of the destination.
    port: # [number] [required] Port - Destination port.
    mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  graphite_output: # [object]
    type: # [string] Output Type
    protocol: # [string] Destination Protocol - Protocol to use when communicating
with the destination.

    # -------------- if protocol is tcp ---------------

    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # ----------------------------------------------------------

    host: # [string] [required] Host - The hostname of the destination.
    port: # [number] [required] Port - Destination port.
    mtu: # [number] Max record Size (Bytes) - Used when Protocol is UDP, to specify
the maximum size of packets sent to the destination. Also known as the MTU for the
network path to the destination system.
    flushPeriodSec: # [number] Flush period (sec) - Used when Protocol is TCP, to
specify how often buffers should be flushed resulting in records sent to the
destination.
```

```
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  router_output: # [object]
    type: # [string] Output Type
    rules: # [array] Rules - Event routing rules
      - filter: # [string] Filter Expression - JavaScript expression to select events to send to output
        output: # [string] Output - Output to send matching events to
        description: # [string] Description - Description of this rule's purpose
        final: # [boolean] Final - Flag to control whether to stop the event from being checked against other rules
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  sqs_output: # [object]
    type: # [string] Output Type
    queueName: # [string] Queue Name - The name, URL, or ARN of the SQS queue to send events to. When a non-AWS URL is specified, format must be: '{url}/myQueueName'. E.g., 'https://host:port/myQueueName'. Must be a JavaScript expression (which can evaluate to a constant value), enclosed in quotes or backticks. Can be evaluated only at init time. E.g., referencing a Global Variable: `https://host:port/myQueue-${C.vars.myVar}`.
    queueType: # [string] [required] Queue Type - The queue type used (or created). Defaults to Standard.
    awsAccountId: # [string] AWS Account ID - SQS queue owner's AWS account ID. Leave empty if SQS queue is in same AWS account.
    messageGroupId: # [string] Message Group ID - This parameter applies only to FIFO queues. The tag that specifies that a message belongs to a specific message group. Messages that belong to the same message group are processed in a FIFO manner. Use event field __messageGroupId to override this value.
    createQueue: # [boolean] Create Queue - Create queue if it does not exist.
    awsAuthenticationMethod: # [string] Authentication method - AWS authentication method. Choose Auto to use IAM roles.

    # -------------- if awsAuthenticationMethod is manual ---------------

    awsApiKey: # [string] Access key - Access key

    # -----------------------------------------------------------


    # -------------- if awsAuthenticationMethod is secret ---------------

    awsSecret: # [string] Secret key pair - Select (or create) a stored secret that references your access key and secret key.

    # -----------------------------------------------------------
    awsSecretKey: # [string] Secret key - Secret key
    region: # [string] Region - AWS Region where the SQS queue is located. Required,
unless the Queue entry is a URL or ARN that includes a Region.
    endpoint: # [string] Endpoint - SQS service endpoint. If empty, defaults to AWS'
Region-specific endpoint. Otherwise, it must point to SQS-compatible endpoint.
    signatureVersion: # [string] Signature version - Signature version to use for
signing SQS requests.
    reuseConnections: # [boolean] Reuse connections - Whether to reuse connections
between requests, which can improve performance.
    rejectUnauthorized: # [boolean] Reject unauthorized certificates - Whether to
reject certificates that cannot be verified against a valid CA (e.g., self-signed
certificates).
    enableAssumeRole: # [boolean] Enable for SQS - Use Assume Role credentials to
access SQS
    assumeRoleArn: # [string] AssumeRole ARN - Amazon Resource Name (ARN) of the
role to assume
    assumeRoleExternalId: # [string] External ID - External ID to use when assuming
role
    maxQueueSize: # [number] Max queue size - Maximum number of queued batches
before blocking.
    maxRecordSizeKB: # [number] Max record size (KB) - Maximum size (KB) of batches
to send. Per the SQS spec, the max allowed value is 256 KB.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max
record size.
    maxInProgress: # [number] Max concurrent requests - The maximum number of in-
progress API requests before backpressure is applied.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  snmp_output: # [object]
```

```
    type: # [string] Output Type
    hosts: # [array] SNMP Trap Destinations - One or more SNMP destinations to
forward traps to
      - host: # [string] Address - Destination host
        port: # [number] Port - Destination port, default is 162
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  sumo_logic_output: # [object]
    type: # [string] Output Type
    url: # [string] API URL - Sumo Logic HTTP collector URL to which events should
be sent.
    customSource: # [string] Custom source name - Optionally, override the source
name configured on the SumoÂ Logic HTTP collector. This can also be overridden at
the event level with the __sourceName field.
    customCategory: # [string] Custom source category - Optionally, override the
source category configured on the SumoÂ Logic HTTP collector. This can also be
overridden at the event level with the __sourceCategory field.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
```

```
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  datadog_output: # [object]
    type: # [string] Output Type
    contentType: # [string] Send logs as - The content type to use when sending
logs.
    message: # [string] Message field - Name of the event field that contains the
message to send. If not specified, Stream sends a JSON representation of the whole
event.
    source: # [string] Source - Name of the source to send with logs. When you send
logs as JSON objects, the event's 'source' field (if set) will override this value.
    host: # [string] Host - Name of the host to send with logs. When you send logs
as JSON objects, the event's 'host' field (if set) will override this value.
    service: # [string] Service - Name of the service to send with logs. When you
send logs as JSON objects, the event's '__service' field (if set) will override this
value.
    tags: # [array of strings] Datadog tags - List of tags to send with logs (e.g.,
'env:prod', 'env_staging:east').
    allowApiKeyFromEvents: # [boolean] Allow API key from events - If enabled, the
API key can be set from the event's '__agent_api_key' field.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    severity: # [string] Severity - Default value for message severity. When you
send logs as JSON objects, the event's '__severity' field (if set) will override
this value.
    site: # [string] Datadog site
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
```

```
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - Organization's API key in Datadog

    # -------------- if authType is manual ---------------


    # ---------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  grafana_cloud_output: # [object]
```

```
    type: # [string] Output Type
    lokiUrl: # [string] Loki URL - The endpoint to send logs to, e.g.: https://logs-prod-us-central1.grafana.net
    prometheusUrl: # [string] [required] Prometheus URL - The remote_write endpoint to send Prometheus metrics to, e.g.: https://prometheus-blocks-prod-us-central1.grafana.net/api/prom/push
    message: # [string] Logs message field - Name of the event field that contains the message to send. If not specified, Stream sends a JSON representation of the whole event.
    messageFormat: # [string] Message Format - Which format to use when sending logs to Loki (Protobuf or JSON).  Defaults to Protobuf.

    # -------------- if messageFormat is json ---------------

    compress: # [boolean] Compress - Whether to compress the payload body before sending. Applies only to Loki's JSON payloads, as both Prometheus' and Loki's Protobuf variant are snappy-compressed by default.

    # --------------------------------------------------------

    labels: # [array] Logs labels - List of labels to send with logs. Labels define Loki streams, so use static labels to avoid proliferating label value combinations and streams. Can be merged and/or overridden by the event's __labels field (e.g.: '__labels: {host: "cribl.io", level: "error"}').
      - name: # [string] Name - Name of the label.
        value: # [string] Value - Value of the label.
    metricRenameExpr: # [string] Metrics renaming expression - A JS expression that can be used to rename metrics. E.g.: name.replace(/\./g, '_') will replace all '.' characters in a metric's name with the supported '_' character. Use the 'name' global variable to access the metric's name.  You can access event fields' values via __e.<fieldName>.
    prometheusAuth: # [object]
      authType: # [string] Authentication Type - The authentication method to use for the HTTP requests

      # -------------- if authType is token ---------------

      token: # [string] Auth token - Bearer token to include in the authorization header. In Grafana Cloud, this is generally built by concatenating the username and the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

      # --------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

      # --------------------------------------------------------


      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for authentication
      password: # [string] Password - Password (a.k.a API key in Grafana Cloud domain) for authentication

      # --------------------------------------------------------
      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret that references your credentials

      # ----------------------------------------------------------

    lokiAuth: # [object]
      authType: # [string] Authentication Type - The authentication method to use for the HTTP requests

      # -------------- if authType is token ---------------

      token: # [string] Auth token - Bearer token to include in the authorization header. In Grafana Cloud, this is generally built by concatenating the username and the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

      # ----------------------------------------------------------


      # -------------- if authType is textSecret ---------------

      textSecret: # [string] Auth token (text secret) - Select (or create) a stored text secret

      # ----------------------------------------------------------


      # -------------- if authType is basic ---------------

      username: # [string] Username - Username for authentication
      password: # [string] Password - Password (a.k.a API key in Grafana Cloud domain) for authentication

      # ----------------------------------------------------------


      # -------------- if authType is credentialsSecret ---------------

      credentialsSecret: # [string] Credentials secret - Select (or create) a secret that references your credentials

      # ----------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking. Warning: Setting this value > 1 can cause Loki and Prometheus to complain about entries being delivered out of order.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body. Warning: Setting this too low can increase the number of ongoing requests (depending on the value of 'Request concurrency'); this can cause Loki and Prometheus to complain about entries being delivered out of order.
    maxPayloadEvents: # [number] Max events per request - Maximum number of events to include in the request body. Default is 0 (unlimited). Warning: Setting this too low can increase the number of ongoing requests (depending on the value of 'Request concurrency'); this can cause Loki and Prometheus to complain about entries being delivered out of order.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Maximum
time between requests. Small values can reduce the payload size below the configured
'Max record size' and 'Max events per request'. Warning: Setting this too low can
increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki and Prometheus to complain about entries being
delivered out of order.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  loki_output: # [object]
    type: # [string] Output Type
    url: # [string] Loki URL - The endpoint to send logs to.
    message: # [string] Logs message field - Name of the event field that contains
the message to send. If not specified, Stream sends a JSON representation of the
whole event.
    messageFormat: # [string] Message Format - Which format to use when sending logs
to Loki (Protobuf or JSON).  Defaults to Protobuf.
```

```
    # -------------- if messageFormat is json ---------------

    compress: # [boolean] Compress - Whether to compress the payload body before
sending.

    # --------------------------------------------------------

    labels: # [array] Logs labels - List of labels to send with logs. Labels define
Loki streams, so use static labels to avoid proliferating label value combinations
and streams. Can be merged and/or overridden by the event's __labels field (e.g.:
'__labels: {host: "cribl.io", level: "error"}').
      - name: # [string] Name - Name of the label.
        value: # [string] Value - Value of the label.
    authType: # [string] Authentication Type - The authentication method to use for
the HTTP requests

    # -------------- if authType is token ---------------

    token: # [string] Auth token - Bearer token to include in the authorization
header. In Grafana Cloud, this is generally built by concatenating the username and
the API key, separated by a colon. E.g.: <your-username>:<your-api-key>.

    # --------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # --------------------------------------------------------


    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for authentication
    password: # [string] Password - Password (a.k.a API key in Grafana Cloud domain)
for authentication

    # --------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # --------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking. Warning: Setting this value > 1 can cause Loki to complain about
entries being delivered out of order.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body. Warning: Setting this too low can increase the number of ongoing
requests (depending on the value of 'Request concurrency'); this can cause Loki to
complain about entries being delivered out of order.
    maxPayloadEvents: # [number] Max events per request - Maximum number of events
to include in the request body. Defaults to 0 (unlimited). Warning: Setting this too
low can increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki to complain about entries being delivered out of
```

```
order.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Maximum
time between requests. Small values can reduce the payload size below the configured
'Max record size' and 'Max events per request'. Warning: Setting this too low can
increase the number of ongoing requests (depending on the value of 'Request
concurrency'); this can cause Loki to complain about entries being delivered out of
order.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ---------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  prometheus_output: # [object]
    type: # [string] Output Type
    url: # [string] Remote Write URL - The endpoint to send metrics to.
    metricRenameExpr: # [string] Metric renaming expression - A JS expression that
can be used to rename metrics. E.g.: name.replace(/\./g, '_') will replace all '.'
```

```
characters in a metric's name with the supported '_' character. Use the 'name'
global variable to access the metric's name.  You can access event fields' values
via __e.<fieldName>.
    sendMetadata: # [boolean] Send metadata - Whether to generate and send metadata
(`type` and `metricFamilyName`) requests.

    # -------------- if sendMetadata is true ---------------

    metricsFlushPeriodSec: # [number] Metadata flush period (sec) - How frequently
metrics metadata is sent out. Value cannot be smaller than the base Flush period
(sec) set above.

    # -------------------------------------------------------

    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
```

```
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication type - Remote Write authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ----------------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # ----------------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # ----------------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  ring_output: # [object]
    type: # [string] Output Type
    format: # [string] Data format - Format of the output data.
    partitionExpr: # [string] Partitioning expression - JS expression to define how
files are partitioned and organized. If left blank, Cribl Stream will fallback on
event.__partition.
    maxDataSize: # [string] Max data size - Maximum disk space allowed to be
consumed (e.g., 420MB or 4GB). Once reached, older data will be deleted.
    maxDataTime: # [string] Max data age - Maximum amount of time to retain data
(e.g., 2h or 4d). Once reached, older data will be deleted.
    compress: # [string] Compression - Select data compression format. Optional.
    destPath: # [string] Path location - Path to use to write metrics. Defaults to
$CRIBL_HOME/state/<id>`
    onBackpressure: # [string] Backpressure behavior - Whether to block or drop
events when all receivers are exerting backpressure.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
```

```
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  open_telemetry_output: # [object]
    type: # [string] Output Type
    endpoint: # [string] Endpoint - The endpoint to send OTEL events to. It can be
any valid URL or an IP address (both IPv4 and IPv6 are supported). If the port is
not specified, it will default to 4317, unless the endpoint is an HTTPS-based URL or
TLS is enabled. In such cases, it will default to 443.
    authType: # [string] Authentication type - OpenTelemetry authentication type

    # -------------- if authType is basic ---------------

    username: # [string] Username - Username for Basic authentication
    password: # [string] Password - Password for Basic authentication

    # ----------------------------------------------------


    # -------------- if authType is token ---------------

    token: # [string] Token - Bearer token to include in the authorization header

    # ----------------------------------------------------


    # -------------- if authType is credentialsSecret ---------------

    credentialsSecret: # [string] Credentials secret - Select (or create) a secret
that references your credentials

    # ----------------------------------------------------


    # -------------- if authType is textSecret ---------------

    textSecret: # [string] Token (text secret) - Select (or create) a stored text
secret

    # ----------------------------------------------------

    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
```

```
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when connecting

      # --------------------------------------------------------

    metadata: # [array] Metadata - Extra information to send with each gRPC request in the form of a list of key-value pairs. Value supports JavaScript expressions that are evaluated just once, when the destination gets started. In case you need to pass credentials as metadata, it's encouraged to use 'C.Secret'.
      - key: # [string] Key - The key of the metadata.
        value: # [string] Value - The value of the metadata.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds) to wait for the connection to establish before retrying
    keepAliveTime: # [number] Keep Alive Time (seconds) - How often the sender should ping the peer to keep the connection alive.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # --------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a
```

```
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  dataset_output: # [object]
    type: # [string] Output Type
    messageField: # [string] Message field - Name of the event field that contains
the message or attributes to send. If not specified, all of the event's non-internal
fields will be sent as attributes.
    excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
event if the Message field is either unspecified or refers to an object. Ignored if
the Message field is a string. If empty, we send all non-internal fields.
    serverHostField: # [string] Server/host field - Name of the event field that
contains the `serverHost` identifier. If not specified, defaults to
`cribl_<outputId>`.
    timestampField: # [string] Timestamp field - Name of the event field that
contains the timestamp. If not specified, defaults to `ts`, `_time`, or
`Date.now()`, in that order.
    defaultSeverity: # [string] Severity - Default value for event severity. If the
`sev` or `__severity` fields are set on an event, the first one matching will
override this value.
    site: # [string] DataSet site - DataSet site to which events should be sent
    customUrl: # [string]
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before
sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
```

```
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter API key directly, or select a
stored secret
    apiKey: # [string] API key - A 'Log Write Access' API key for the DataSet
account

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] API key (text secret) - Select (or create) a stored text
secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  logstream_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    host: # [string] Address - The hostname of the receiver
    port: # [number] Port - The port to connect to on the provided host

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    hosts: # [array] Destinations - Set of hosts to load-balance data to.
      - host: # [string] Address - The hostname of the receiver.
        port: # [number] Port - The port to connect to on the provided host.
        tls: # [string] TLS - Whether to inherit TLS configs from group setting or
disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a
```

```
TLS connection. If not specified, defaults to connection host (iff not an IP);
otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent
connections (per worker process). A random set of IPs will be picked on every DNS
resolution period. Use 0 for unlimited.

    # ----------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before
sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to
throttle while writing to an output. Also takes values with multiple-byte units,
such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ----------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds)
to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait
for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
```

```
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    authType: # [string] Authentication method - Enter a token directly, or provide
a secret referencing a token
    authToken: # [string] Auth token - Optional authentication token to include as
part of the connection header

    # -------------- if authType is manual ---------------


    # ----------------------------------------------------------

    textSecret: # [string] Auth token (text secret) - Select (or create) a stored
text secret

    # -------------- if authType is secret ---------------


    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
  cribl_tcp_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    host: # [string] Address - The hostname of the receiver
    port: # [number] Port - The port to connect to on the provided host

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    hosts: # [array] Destinations - Set of hosts to load-balance data to.
      - host: # [string] Address - The hostname of the receiver.
        port: # [number] Port - The port to connect to on the provided host.
        tls: # [string] TLS - Whether to inherit TLS configs from group setting or disable TLS.
        servername: # [string] TLS Servername - Servername to use if establishing a TLS connection. If not specified, defaults to connection host (iff not an IP); otherwise, to the global TLS settings.
        weight: # [number] Load Weight - The weight to use for load-balancing purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How far back in time to keep traffic stats for load balancing purposes.
    maxConcurrentSenders: # [number] Max connections - Maximum number of concurrent connections (per worker process). A random set of IPs will be picked on every DNS resolution period. Use 0 for unlimited.

    # ----------------------------------------------------------

    compression: # [string] Compression - Codec to use to compress the data before sending
    throttleRatePerSec: # [string] Throttling - Rate (in bytes per second) to throttle while writing to an output. Also takes values with multiple-byte units, such as KB, MB, GB, etc. (E.g., 42 MB.) Default value of 0 specifies no throttling.
    tls: # [object] TLS settings (client side)
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when connecting

      # ----------------------------------------------------------

    connectionTimeout: # [number] Connection Timeout - Amount of time (milliseconds) to wait for the connection to establish before retrying
    writeTimeout: # [number] Write Timeout - Amount of time (milliseconds) to wait for a write to complete before assuming connection is dead
    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes before the internally generated authentication token expires, valid values between 1 and 60
    excludeFields: # [array of strings] Exclude Fields - Fields to exclude from the event. By default, all internal fields except `__output` are sent.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.
    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to this output.
    systemFields: # [array of strings] System fields - Set of fields to automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards supported.
    environment: # [string] Environment - Optionally, enable this config only on a specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in Stream.
  cribl_http_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    url: # [string] Cribl endpoint - URL of a Cribl Worker to send events to, e.g., http://localhost:10200
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS lookup. When a DNS server returns multiple addresses, this will cause Stream to cycle through them in the order returned.

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the current host from the list of any resolved hostnames.
    urls: # [array] Cribl Worker Endpoints
      - url: # [string] Cribl Endpoint - URL of a Cribl Worker to send events to, e.g., http://localhost:10200
        weight: # [number] Load Weight - The weight to use for load-balancing purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How far back in time to keep traffic stats for load balancing purposes.

    # ----------------------------------------------------------

    tls: # [object] TLS settings (client side)
```

```
      disabled: # [boolean] Disabled

      # -------------- if disabled is false ---------------

      rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to No.
      servername: # [string] Server name (SNI) - Server name for the SNI (Server
Name Indication) TLS extension. It must be a host name, and not an IP address.
      certificateName: # [string] Certificate name - The name of the predefined
certificate.
      caPath: # [string] CA certificate path - Path on client in which to find CA
certificates to verify the server's cert. PEM format. Can reference $ENV_VARS.
      privKeyPath: # [string] Private key path (mutual auth) - Path on client in
which to find the private key to use. PEM format. Can reference $ENV_VARS.
      certPath: # [string] Certificate path (mutual auth) - Path on client in which
to find certificates to use. PEM format. Can reference $ENV_VARS.
      passphrase: # [string] Passphrase - Passphrase to use to decrypt private key.
      minVersion: # [string] Minimum TLS version - Minimum TLS version to use when
connecting
      maxVersion: # [string] Maximum TLS version - Maximum TLS version to use when
connecting

      # -------------------------------------------------------

    tokenTTLMinutes: # [number] Auth Token TTL minutes - The number of minutes
before the internally generated authentication token expires, valid values between 1
and 60.
    excludeFields: # [array of strings] Exclude fields - Fields to exclude from the
event. By default, all internal fields except `__output` are sent.
    compression: # [string] Compression - Codec to use to compress the data before
sending.
    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the
request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to
include in the request body. Default is 0 (unlimited).
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are
not authorized by a CA in the CA certificate path, or by another trusted CA (e.g.,
the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for
a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests.
Small values could cause the payload size to be smaller than the configured Max body
size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines
which data should be logged when a request fails. Defaults to None.  All headers are
redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe
to log in plain text.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or
queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------
    pqMaxFileSize: # [string] Max file size - The maximum size to store in each
queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the
queue is allowed to consume. Once reached, the system stops queueing and applies the
fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue
files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted
data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop
events when the queue is exerting backpressure (full capacity or low disk). 'Block'
is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data,
while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
  humio_hec_output: # [object]
    type: # [string] Output Type
    loadBalanced: # [boolean] Load balancing - Use load-balanced destinations

    # -------------- if loadBalanced is false ---------------

    url: # [string] Humio HEC Endpoint - URL to a Humio HEC endpoint to send events
to, e.g., https://cloud.us.humio.com/api/v1/ingest/hec for JSON and
https://cloud.us.humio.com/api/v1/ingest/hec/raw for raw
    useRoundRobinDns: # [boolean] Round-robin DNS - Enable to use round-robin DNS
lookup. When a DNS server returns multiple addresses, this will cause Stream to
cycle through them in the order returned.

    # ----------------------------------------------------------


    # -------------- if loadBalanced is true ---------------

    excludeSelf: # [boolean] Exclude current host IPs - Exclude all IPs of the
current host from the list of any resolved hostnames.
    urls: # [array] Humio HEC Endpoints
      - url: # [string] HEC Endpoint - URL to a Humio HEC endpoint to send events
to, e.g., https://cloud.us.humio.com/api/v1/ingest/hec for JSON and
https://cloud.us.humio.com/api/v1/ingest/hec/raw for raw
        weight: # [number] Load Weight - The weight to use for load-balancing
purposes.
    dnsResolvePeriodSec: # [number] DNS resolution period (seconds) - Re-resolve any
hostnames every this many seconds and pick up destinations from A records.
    loadBalanceStatsPeriodSec: # [number] Load balance stats period (seconds) - How
far back in time to keep traffic stats for load balancing purposes.

    # ----------------------------------------------------------

    concurrency: # [number] Request concurrency - Maximum number of ongoing requests
before blocking.
```

```
    maxPayloadSizeKB: # [number] Max body size (KB) - Maximum size, in KB, of the request body.
    maxPayloadEvents: # [number] Max events per request - Max number of events to include in the request body. Default is 0 (unlimited).
    compress: # [boolean] Compress - Whether to compress the payload body before sending.
    rejectUnauthorized: # [boolean] Validate server certs - Reject certs that are not authorized by a CA in the CA certificate path, or by another trusted CA (e.g., the system's CA). Defaults to Yes.
    timeoutSec: # [number] Request timeout - Amount of time, in seconds, to wait for a request to complete before aborting it.
    flushPeriodSec: # [number] Flush period (sec) - Maximum time between requests. Small values could cause the payload size to be smaller than the configured Max body size.
    extraHttpHeaders: # [array] Extra HTTP headers - Extra HTTP headers.
      - name: # [string] Name - Field name
        value: # [string] Value - Field value
    failedRequestLoggingMode: # [string] Failed request logging mode - Determines which data should be logged when a request fails. Defaults to None.  All headers are redacted by default, except those listed under `Safe Headers`.
    safeHeaders: # [array of strings] Safe headers - List of headers that are safe to log in plain text.
    format: # [string] Request Format - Send data in JSON format to the api/v1/ingest/hec endpoint , or raw 1-request-per-line to the api/v1/ingest/hec/raw endpoint .
    authType: # [string] Authentication method - Enter a token directly, or provide a secret referencing a token

    # -------------- if authType is manual ---------------

    token: # [string] HEC Auth token - Humio HEC authentication token

    # ----------------------------------------------------------


    # -------------- if authType is secret ---------------

    textSecret: # [string] HEC Auth token (text secret) - Select (or create) a stored text secret

    # ----------------------------------------------------------

    onBackpressure: # [string] Backpressure behavior - Whether to block, drop, or queue events when all receivers are exerting backpressure.

    # -------------- if onBackpressure is queue ---------------

    pqMaxFileSize: # [string] Max file size - The maximum size to store in each queue file before closing and optionally compressing (KB, MB, etc.).
    pqMaxSize: # [string] Max queue size - The maximum amount of disk space the queue is allowed to consume. Once reached, the system stops queueing and applies the fallback Queue-full behavior. Enter a numeral with units of KB, MB, etc.
    pqPath: # [string] Queue file path - The location for the persistent queue files. To this field's value, the system will append: /<worker-id>/<output-id>.
    pqCompress: # [string] Compression - Codec to use to compress the persisted data.
    pqOnBackpressure: # [string] Queue-full behavior - Whether to block or drop events when the queue is exerting backpressure (full capacity or low disk). 'Block' is the same behavior as non-PQ blocking. 'Drop new data' throws away incoming data, while leaving the contents of the PQ unchanged.
    pqControls: # [object]

    # ----------------------------------------------------------

    pipeline: # [string] Pipeline - Pipeline to process data before sending out to
this output.
    systemFields: # [array of strings] System fields - Set of fields to
automatically add to events using this output. E.g.: cribl_pipe, c*. Wildcards
supported.
    environment: # [string] Environment - Optionally, enable this config only on a
specified Git branch. If empty, will be enabled everywhere.
    streamtags: # [array of strings] Tags - Add tags for filtering and grouping in
Stream.
```

;

10.6.15. 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, in order. If not specified parser will auto-generate.
```

;

10.6.16. policies.yml

`policies.yml` contains RBAC Policy definitions. Default example:

$CRIBL_HOME/default/cribl/policies.yml

```
GroupFull:
  args:
    - groupName
  template:
    - PATCH /master/groups/${groupName}/deploy
    - GroupEdit ${groupName}
GroupEdit:
  args:
    - groupName
  template:
    - '* /m/${groupName}'
    - '* /m/${groupName}/*'
    - GroupRead ${groupName}
GroupCollect:
  args:
    - groupName
  template:
    - POST /m/${groupName}/lib/jobs
    - PATCH /m/${groupName}/lib/jobs/*
    - POST /m/${groupName}/jobs
    - PATCH /m/${groupName}/jobs/*
    - GroupRead ${groupName}
GroupRead:
  args:
    - groupName
  template:
    - GET /m/${groupName}
    - GET /m/${groupName}/*
    - POST /m/${groupName}/preview
    - POST /m/${groupName}/system/capture
    - POST /m/${groupName}/lib/expression
    - GET /master/groups/${groupName}
    - GET /master/workers
    - GET /master/workers/*
    - '* /w/*'
    - GET /master/groups
    - GET /system/info
    - GET /system/info/*
    - GET /system/logs
    - GET /system/logs/group/${groupName}/*
    - GET /system/settings
    - GET /system/settings/*
    - GET /system/instance/distributed
    - GET /system/instance/distributed/*
    - GET /version
    - GET /version/*
    - GET /version/info
    - GET /version/info/*
    - GET /version/status
    - GET /version/status/*
    - GET /mappings
    - GET /mappings/*
    - GET /system/messages
    - GET /system/message/*
    - GET /ui/*
    - POST /system/metrics/query
    - GET /clui
    - POST /system/capture
```

;

10.6.17. regexes.yml

`regexes.yml` maintains a list of regexes. Cribl's Regex Library ships under `default`. Each regex is listed according to the following pattern:

$CRIBL_HOME/default/cribl/regexes.yml

```
egex_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this regex. Optional.
  regex: # [string] Regex pattern - Regex pattern. Required.
  sampleData: # [string] Sample data - Sample data for this regex. Optional.
  tags: # [string] Tags - One or more tags related to this regex. Optional.
```

;

10.6.18. roles.yml

`roles.yml` contains RBAC Role definitions. Default example:

$CRIBL_HOME/default/cribl/roles.yml

```
admin:
  description: 'Members with admin role have permission to do anything and
everything in the system.'
  policy:
    - '* *'
reader_all:
  description: 'Members with reader_all role get read-only access to all
Worker Groups/Fleets.'
  policy:
    - GroupRead *
collect_all:
  description: 'Members of this group can run existing collection jobs of all
Worker Groups/Fleets'
  policy:
    - GroupCollect *
editor_all:
  description: 'Members with editor_all role get read/write access to all
Worker Groups/Fleets.'
  policy:
    - GroupEdit *
owner_all:
  description: 'Members with owner_all role get read/write access as well as Deploy
permissions to all Worker Groups/Fleets.'
  policy:
    - GroupFull *
user:
  description: 'The base user role allows users to see the system info along with
their own profile settings.'
  policy:
    - GET /system/info
    - GET /system/info/*
    - GET /system/users
    - GET /system/instance/distributed
    - GET /system/instance/distributed/*
    - GET /clui
    - PATCH /ui/*
```

;

10.6.19. samples.yml

`samples.yml` contains metadata about about stored sample data files (size, number of events, date created, name, etc.). Each sample is listed according to the following pattern:

$CRIBL_HOME/local/cribl/samples.yml

```yaml
sample_id: # [object]
  sampleName: # [string] File Name - Filename to save the sample as. Required.
  pipelineId: # [string] Associate with Pipeline - Select a pipeline to associate with sample with. Select GLOBAL if not sure. Deprecated.
  description: # [string] Description - Brief description of this sample file. Optional.
  ttl: # [number] Expiration (hours) - Time to live for the sample, the TTL is reset after each use of the sample. Leave empty to never expire.
  tags: # [string] Tags - One or more tags related to this sample file. Optional.
```

The corresponding sample files reside in `$CRIBL_HOME/data/samples`.

;

10.6.20. schemas.yml

`schemas.yml` stores configuration data for the Knowledge > Schema Library.

$CRIBL_HOME/default/cribl/schemas.yml

```
schema_id: # [object]
  description: # [string] Description - Brief description of this schema. Optional.
  schema: # [string] Schema - JSON schema matching standards of draft version 2019-09.
```

;

10.6.21. scripts.yml

`scripts.yml` stores configuration data for scripts configured at global ⚙ **Settings** (lower left) > **Scripts**:

$CRIBL_HOME/local/cribl/scripts.yml

```
script_id: # [object]
  command: # [string] Command - Command to execute for this script
  description: # [string] Description - Brief description of this script. Optional.
  args: # [array of strings] Arguments - Arguments to pass when executing this script
  env: # [object] Env Variables - Extra environment variables to set when executing script
```

;

10.6.22. vars.yml

`vars.yml` stores configuration data for the Knowledge > Global Variables Library.

$CRIBL_HOME/default/cribl/vars.yml

```yaml
variable_id: # [object]
  lib: # [string] Library
  description: # [string] Description - Brief description of this variable.
Optional.
  type: # [string] Type - Type of variable.
  value: # [string] Value - Value of variable
  tags: # [string] Tags - One or more tags related to this variable. Optional.
```

;

11. Techniques & Tips

11.1. 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

In Cribl Stream 3.5 and above, 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.

;

11.2. Integrating with Other Services

11.2.1. 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.

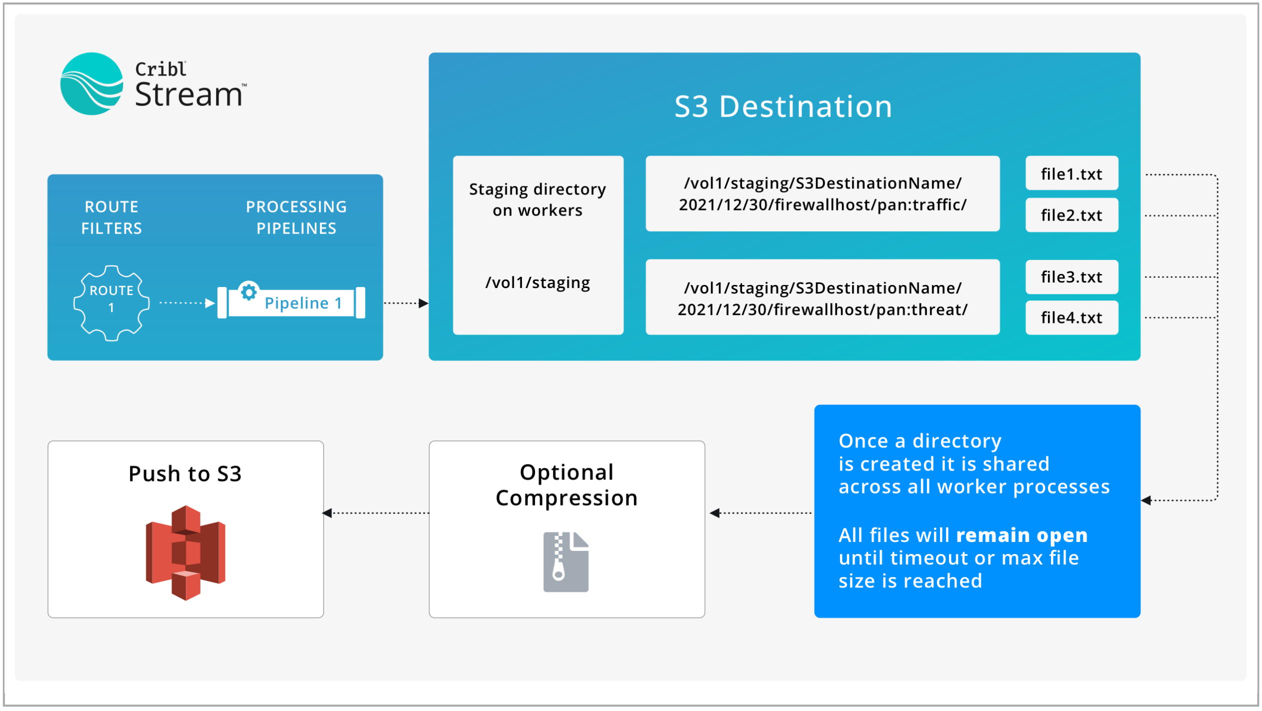
Then, this guide will go on to detail 10 "better practices" (and a bonus one!) as we cover four broad topics:

- 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
ROUTE FILTERS
PROCESSING PIPELINES
ROUTE 1
Pipeline 1
S3 Destination
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, there are two main things to consider:

- **File 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 the oldest 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.

  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.

- **Partitioning Expression**: To limit the number of files in each directory, partitioning expressions 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 Destination. 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.
- 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. Try updating the Worker's Linux system settings to accommodate a larger quantity of open files. When setting the maximum open files substantially higher than 65,536, you should track system health.

Cribl recommends setting the maximum open files based on the number of S3 Destinations you are planning, 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
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 first group of settings. You will be required to reboot.

```
# sysctl -w fs.file-max=65536 (as root)

# vi /etc/sysctl.conf (as root)
fs.file-max = 65536

# 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. For every combination of partitioning expression, file prefix expression, and Worker Process, 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}/${host}/${sourcetype}
   ```

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.

```
`${C.Time.strftime(_time ? _time : Date.now(), '%Y/%m/%d/%H')}
/${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.

```
`${C.Time.strftime(_time ? _time : Date.now(),
'%Y/%m/%d/%H')}/${state}/${interface}/${dest_ip}`
```

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!

```
`${C.Time.strftime(_time ? _time : Date.now(),
'%Y/%m/%d/%H')}/${state}/${interface}/${src_ip}`
```

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
```

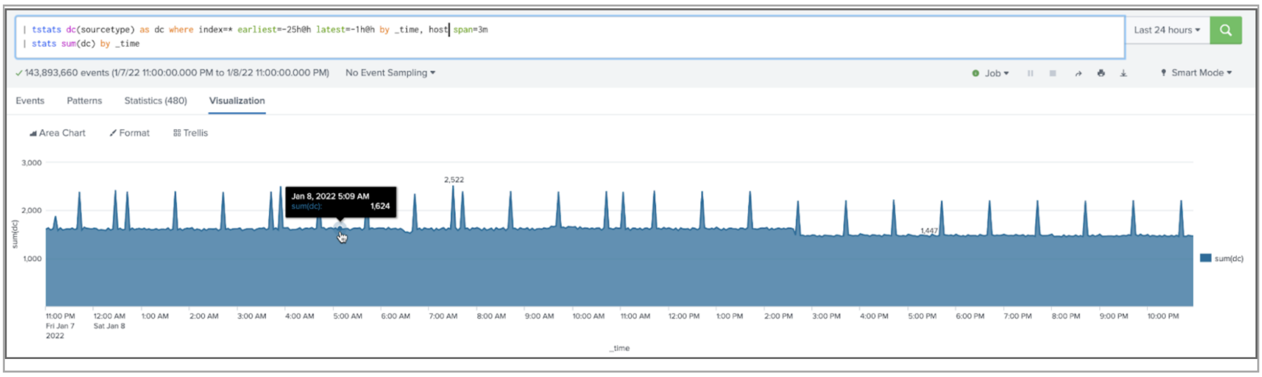
Here's an example using `tstats`:

```
| tstats dc(source) as dc where index=* earliest=-25h@h latest=-1h@h by _time, host,
sourcetype span=3m
| 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) No Event Sampling ▾

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 or TB/day) you expect to ingest, times `1.25` metadata factor; divided by
- `1440` (minutes in a day) times `Duration` (hours) 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.

$$\frac{\mathit{Data\ Volume\ (GB/day)} \times 1.25\ \mathit{metadata\ factor}}{1440\ \mathit{mins/day}} \times \mathit{mins\ to\ store\ data\ to\ disk} \div \mathit{number\ of\ Workers}$$

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-logstream-assumerole-role"
    },
    "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" --role-session-name AWSCLI-Session
```

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
```

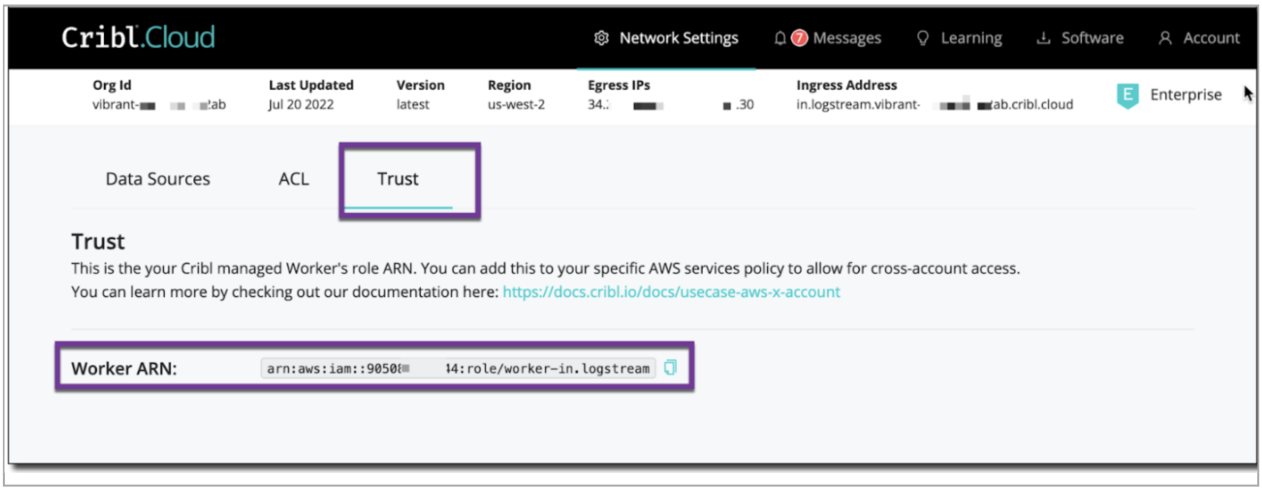
Example of reading a file from S3:

```
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.



Org Id	Last Updated	Version	Region	Egress IPs	Ingress Address
vibrant-[redacted]ab	Jul 20 2022	latest	us-west-2	34.[redacted].30	in.logstream.vibrant-[redacted]ab.cribl.cloud

Trust

This is the your Cribl managed Worker's role ARN. 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::9050[redacted]4:role/worker-in.logstream`

Cribl.Cloud ARN

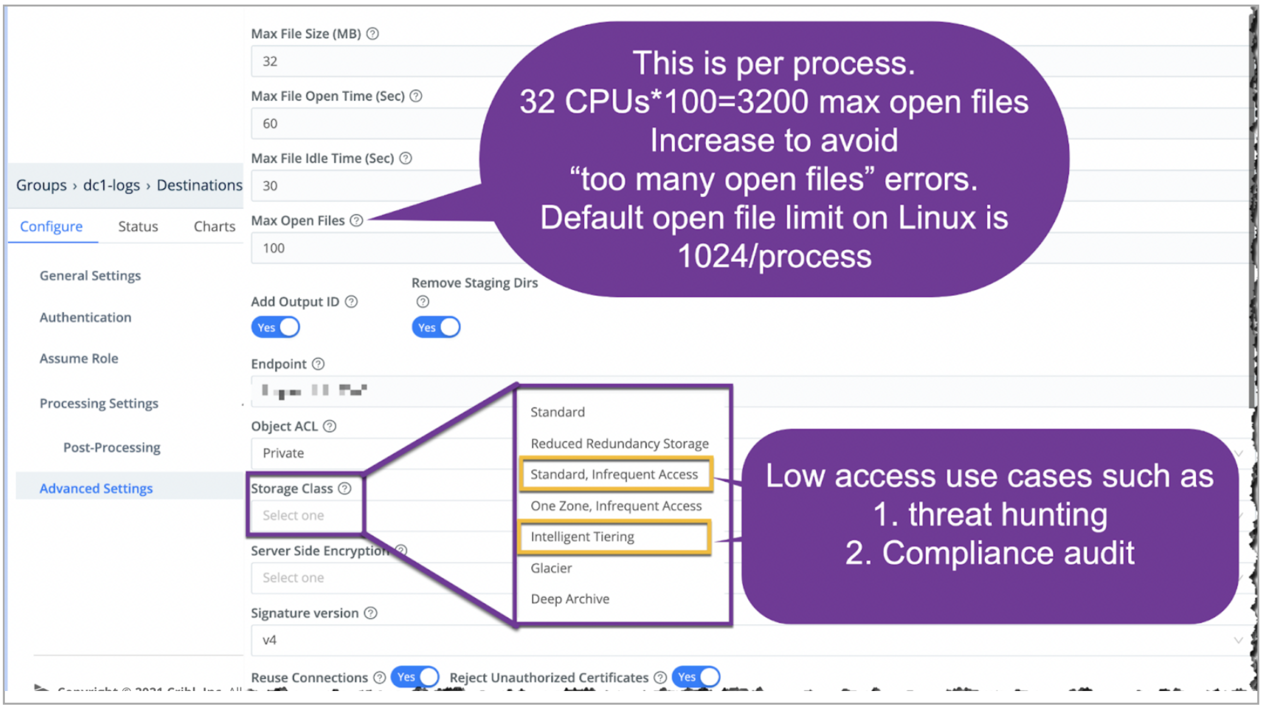
Configure your AWS account with access to the S3 bucket, to allow Cribl.Cloud to assume that role, as illustrated above in Amazon S3 Permissions.

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.
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. 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.
8. **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.
9. **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.



Max File Size (MB): 32

Max File Open Time (Sec): 60

Max File Idle Time (Sec): 30

Max Open Files: 100

Add Output ID: Yes

Remove Staging Dirs: Yes

Endpoint

Object ACL: Private

Storage Class: Select one

Server Side Encryption: Select one

Signature version: v4

Reuse Connections: Yes

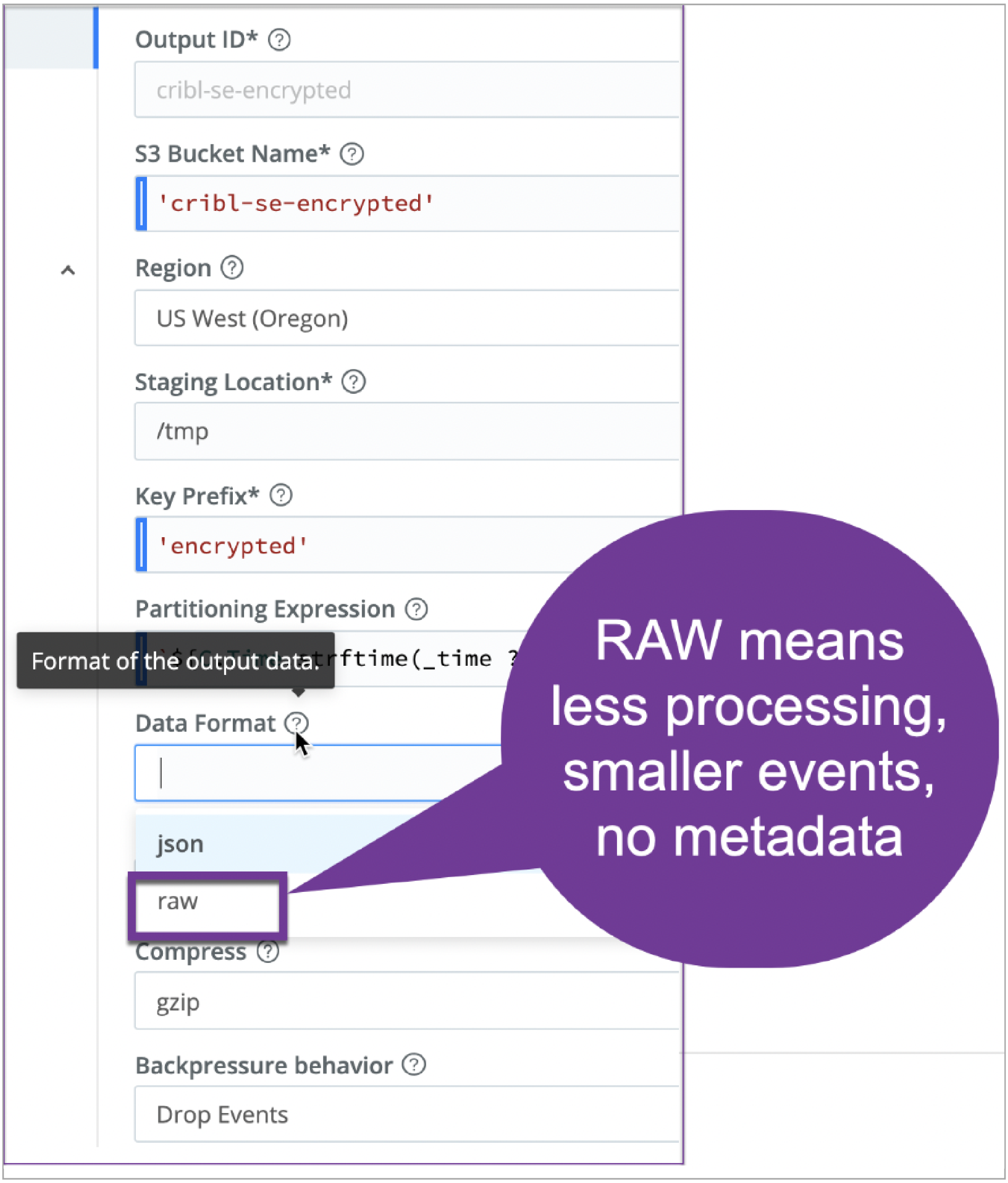
Reject Unauthorized Certificates: Yes

Standard
Reduced Redundancy Storage
Standard, Infrequent Access
One Zone, Infrequent Access
Intelligent Tiering
Glacier
Deep Archive

This is per process.
32 CPUs*100=3200 max open files
Increase to avoid
“too many open files” errors.
Default open file limit on Linux is 1024/process

Low access use cases such as
1. threat hunting
2. Compliance audit

Configuring S3 Destination Settings



Output ID* cribl-se-encrypted

S3 Bucket Name* 'cribl-se-encrypted'

Region US West (Oregon)

Staging Location* /tmp

Key Prefix* 'encrypted'

Partitioning Expression `${C.Time.strftime(_time ?

Format of the output data.

Data Format

json

raw

Compress gzip

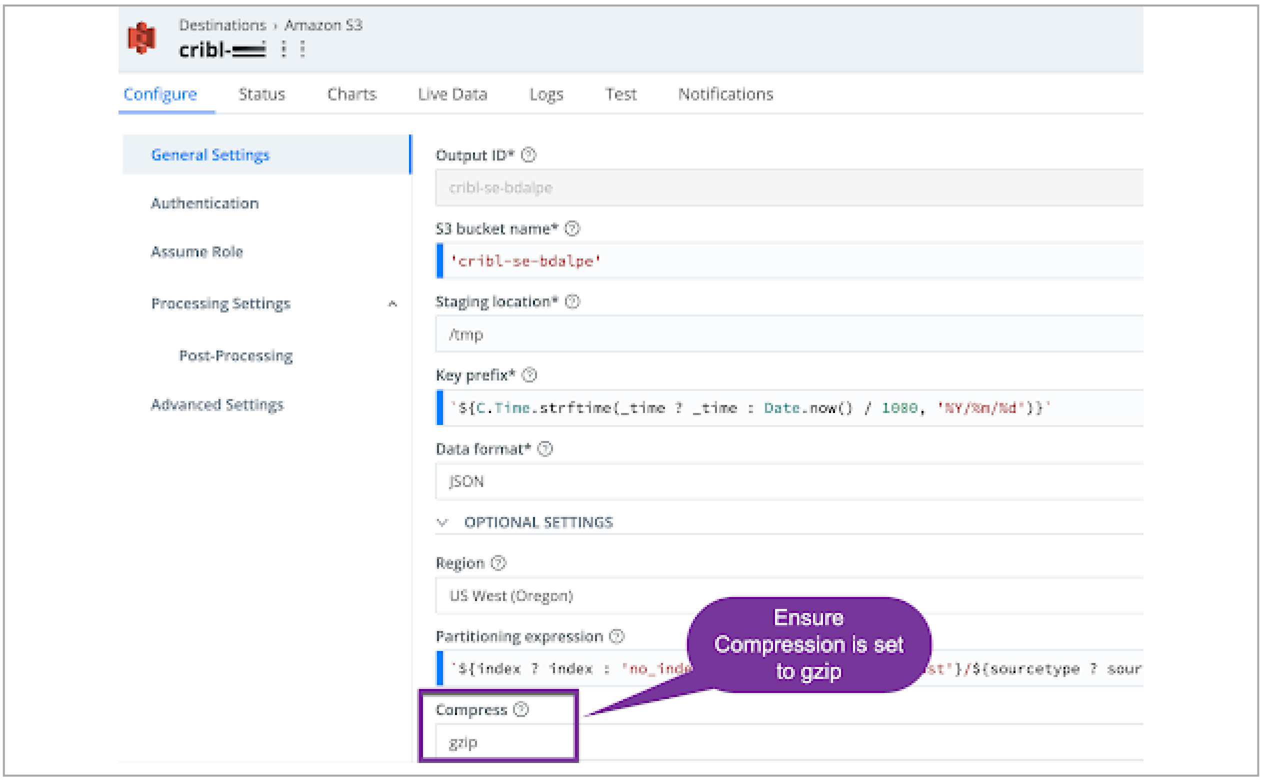
Backpressure behavior Drop Events

RAW means less processing, smaller events, no metadata

Configuring S3 Destination Settings- Continued

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. This will reduce both egress costs and S3 storage costs.



Destinations › Amazon S3

cribl-[redacted]

Output ID*
cribl-se-bdalpe

S3 bucket name*
'cribl-se-bdalpe'

Staging location*
/tmp

Key prefix*
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%Y/%m/%d')}`

Data format*
JSON

OPTIONAL SETTINGS

Region
US West (Oregon)

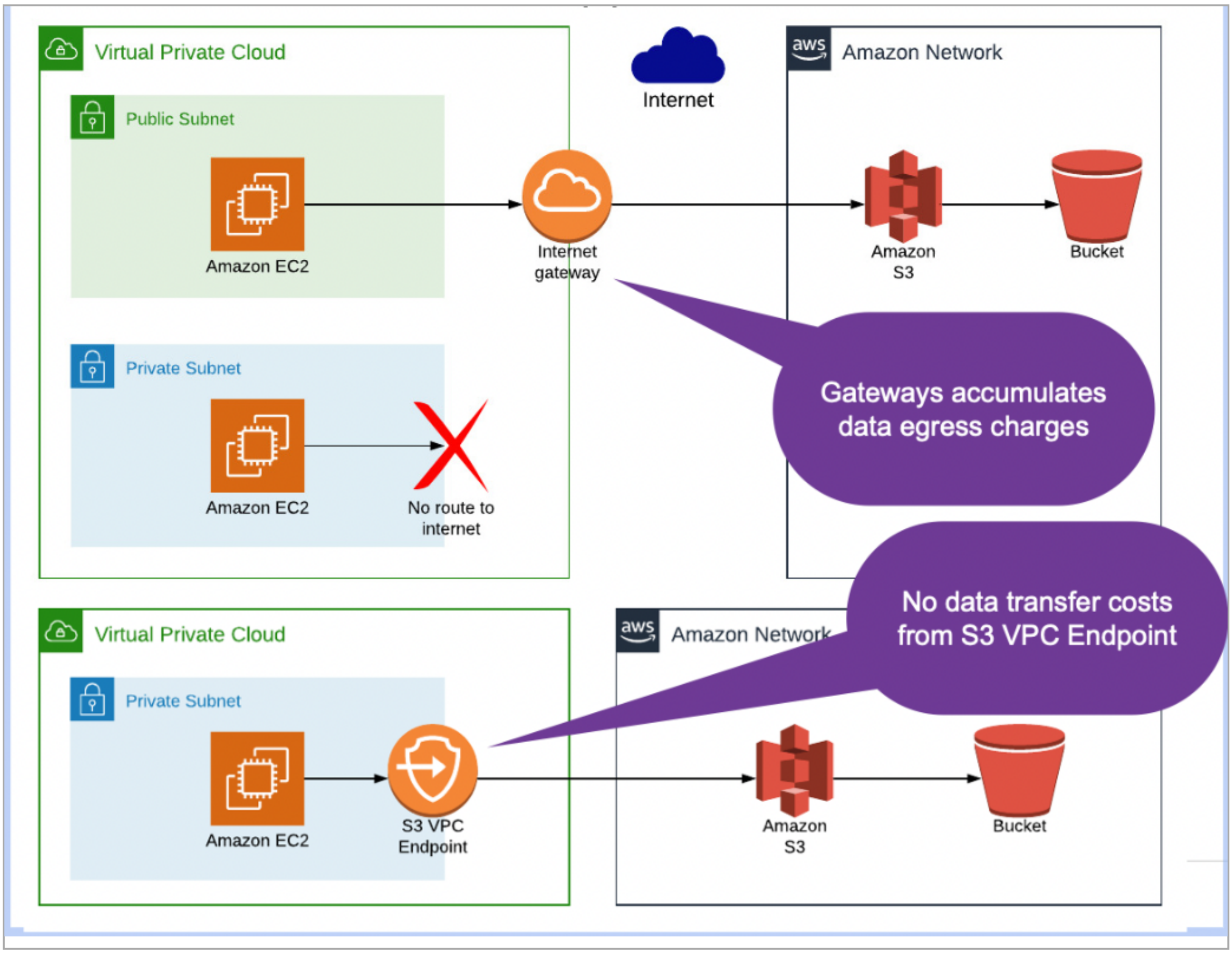
Partitioning expression
`${index ? index : 'no_inde st'}/${sourcetype ? sour

Compress
gzip

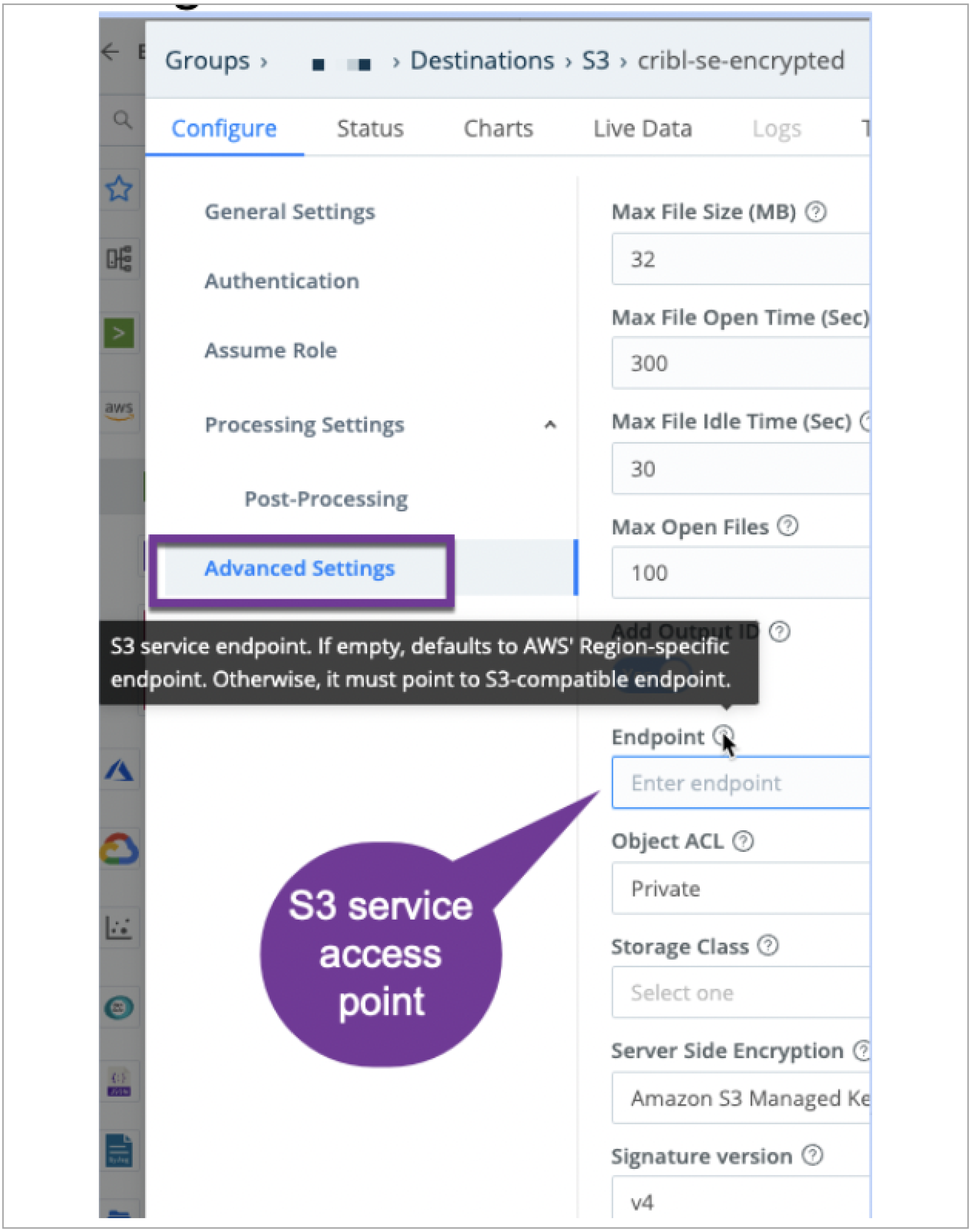
Ensure Compression is set to gzip

Set File Compression

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



Groups › Destinations › S3 › cribl-se-encrypted

Max File Size (MB): 32

Max File Open Time (Sec): 300

Max File Idle Time (Sec): 30

Max Open Files: 100

Add Output ID

S3 service endpoint. If empty, defaults to AWS' Region-specific endpoint. Otherwise, it must point to S3-compatible endpoint.

Endpoint: Enter endpoint

Object ACL: Private

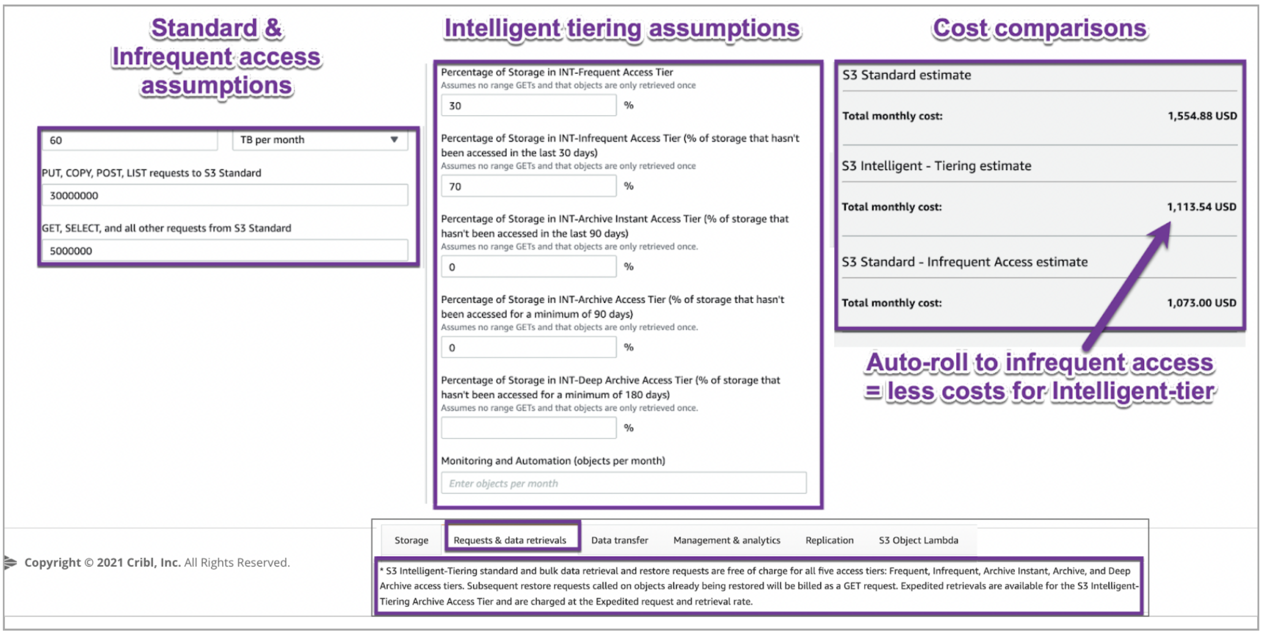
Storage Class: Select one

Server Side Encryption: Amazon S3 Managed Ke

Signature version: v4

VPC Endpoint URL in the Cribl S3 Destination

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.

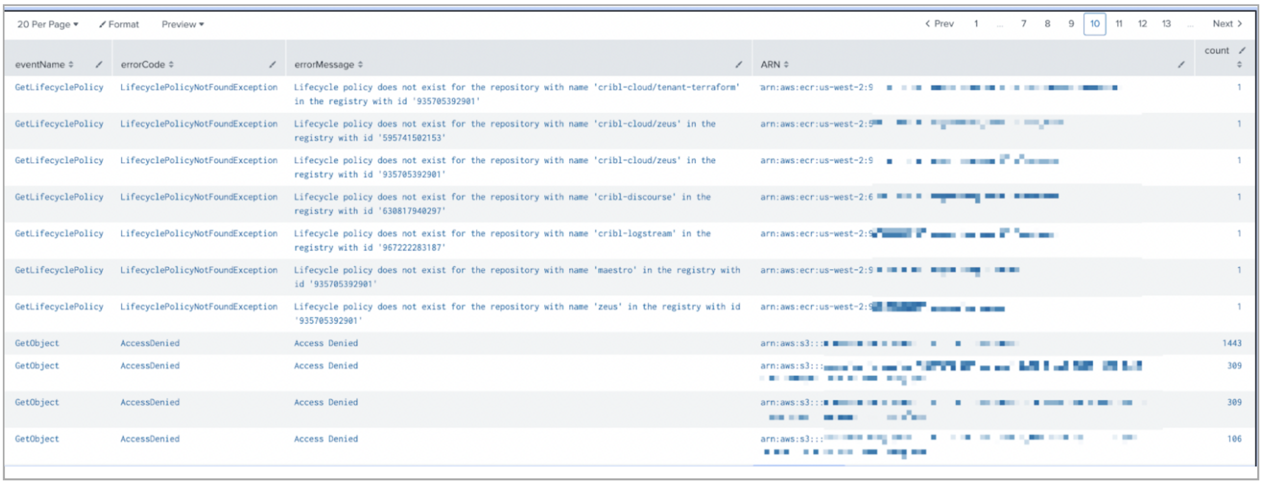
AWS Cost Comparison

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:[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:[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:[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:[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:[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:[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:[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

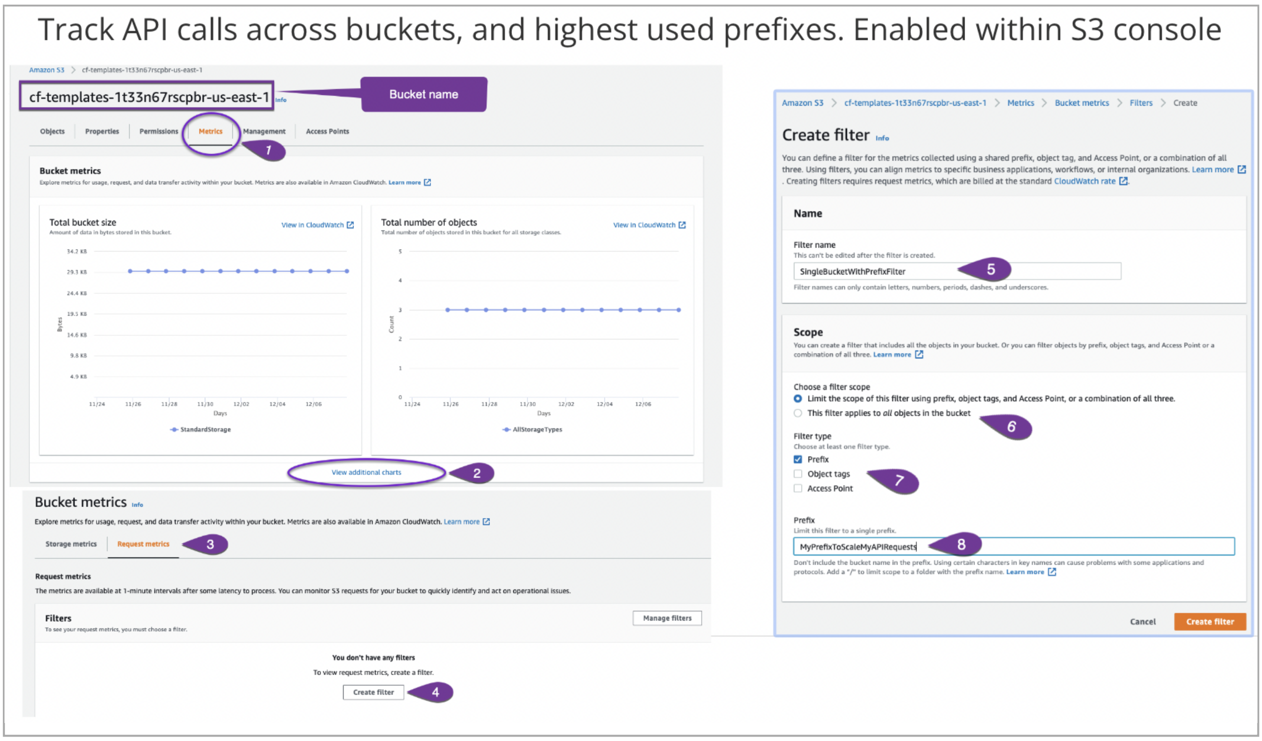
If there are `rate exceeded` messages from writing to Amazon S3, then a new strategy for partition and filename prefix expressions might be warranted. You will want to create fewer files, so consider one or more of the following options:

- 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

cf-templates-1t33n67rscpbr-us-east-1

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

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.

Create filter

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

This can't be edited after the filter is created.

SingleBucketWithPrefixFilter

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

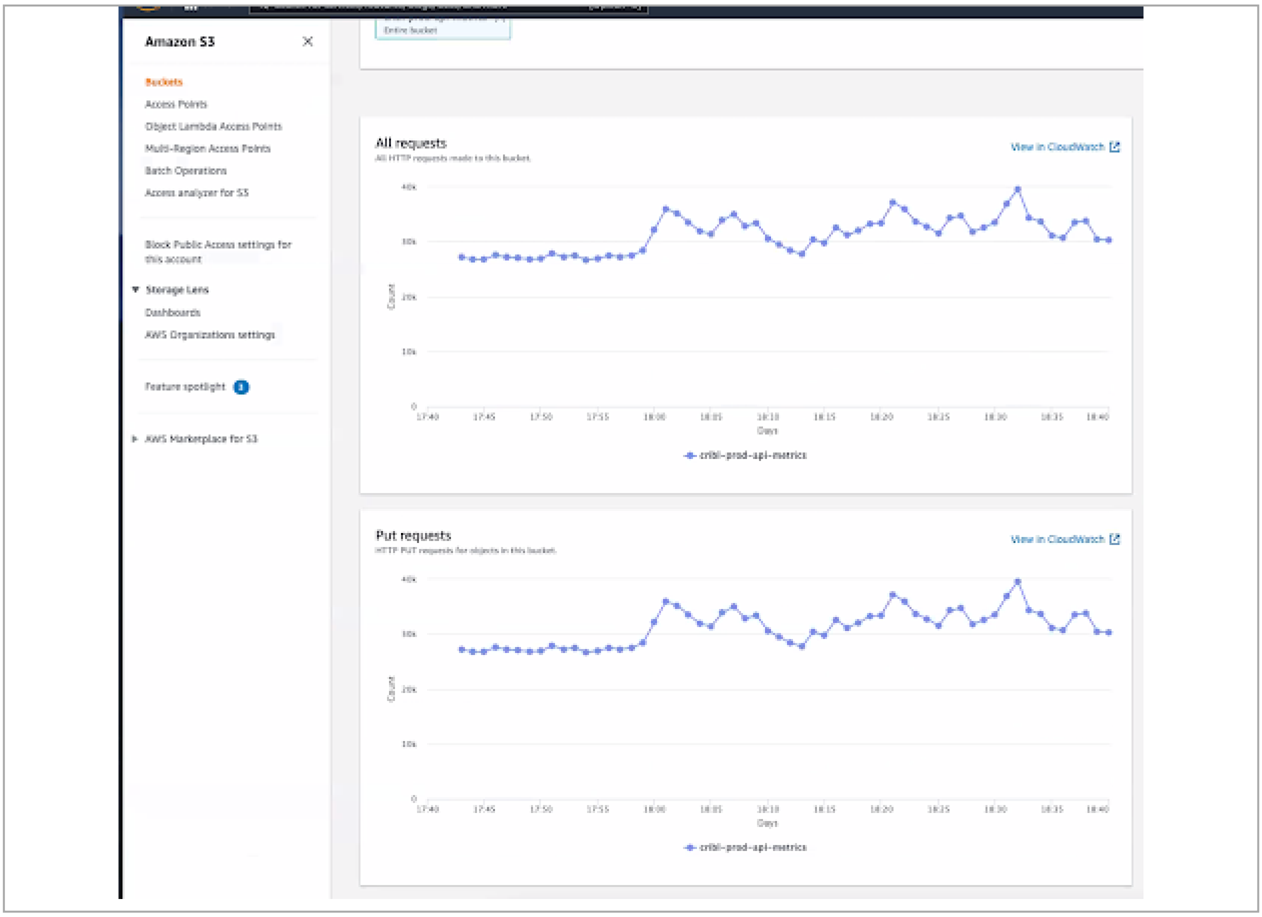
Limit this filter to a single prefix.

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:

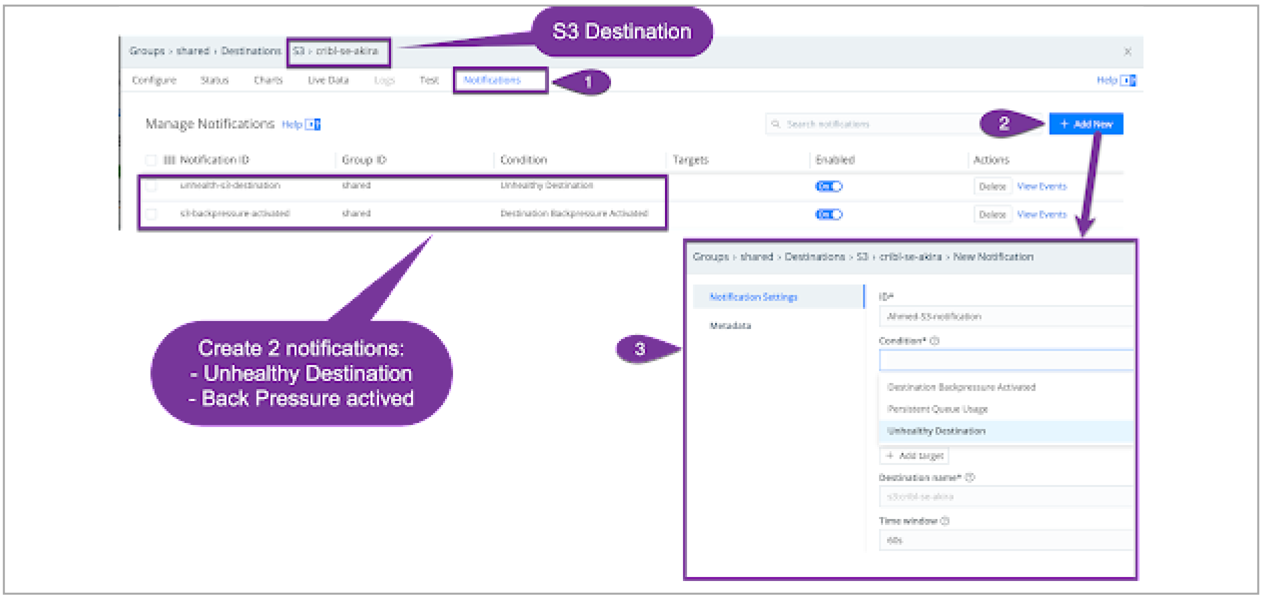
- `BurstBalance`
- `VolumeReadBytes`
- `VolumeWriteBytes`
- `VolumeReadOps`
- `VolumeWriteOps`
- `VolumeQueueLength`

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, the screenshot below illustrates the workflow, which includes:

- Editing Cribl Stream's S3 Destination to configure Notifications.
- Adding Notifications for different conditions.



Manage Notifications

Notification ID	Group ID	Condition	Targets	Enabled	Actions
unhealth-s3-destination	shared	Unhealthy Destination		On	Delete View Events
s3-backpressure-activated	shared	Destination Backpressure Activated		On	Delete View Events

ID*: Ahmed-S3-notification

Condition*:
- Destination Backpressure Activated
- Persistent Queue Usage
- Unhealthy Destination

Destination name*: s3:cribl-se-akira

Time window: 60s

S3 Destination Notifications

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 queues 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.

Better 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(), '%Y/%m/%d/%H/%M')}
```

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 * 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

```
${C.Time.strftime(_time ? _time : Date.now(), '%Y/%m/%d/%H')}
/${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?	30 * 1 = 30 files
What if there are 10 Nodes in the Worker Group? How many files in each S3 prefix?	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(), '%Y/%m/%d/%H')}
```

KEY QUESTIONS	ANSWERS

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?	Over API limit of 5,500 reads/sec.

Result

- A bad partitioning expression.

Solution

- Go down to the minute, to lower the number of files in any one AWS prefix.

```
${C.Time.strftime(_time ? _time : Date.now(), '%Y/%m/%d/%H/%M')}
```

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.

;

11.2.2. 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.

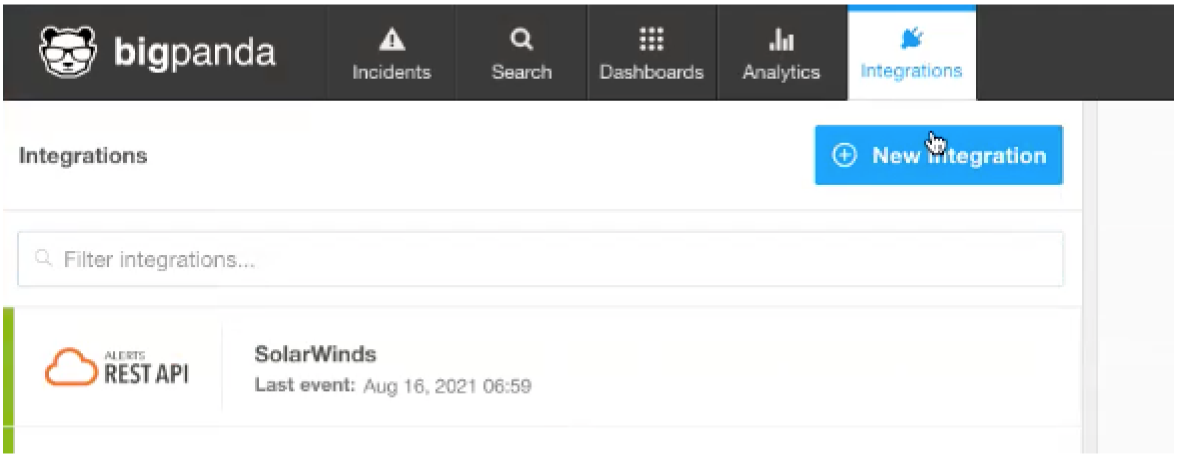
Before you begin, you should have an Admin account on a BigPanda Cloud instance.

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.)

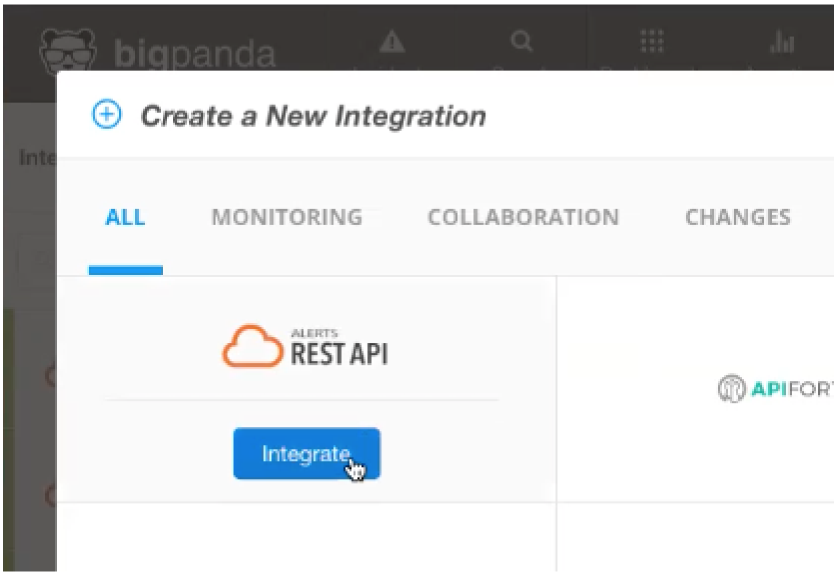
1. Log into your BigPanda Cloud instance as an Admin.

2. In the **Integrations** tab, click **New Integration**.



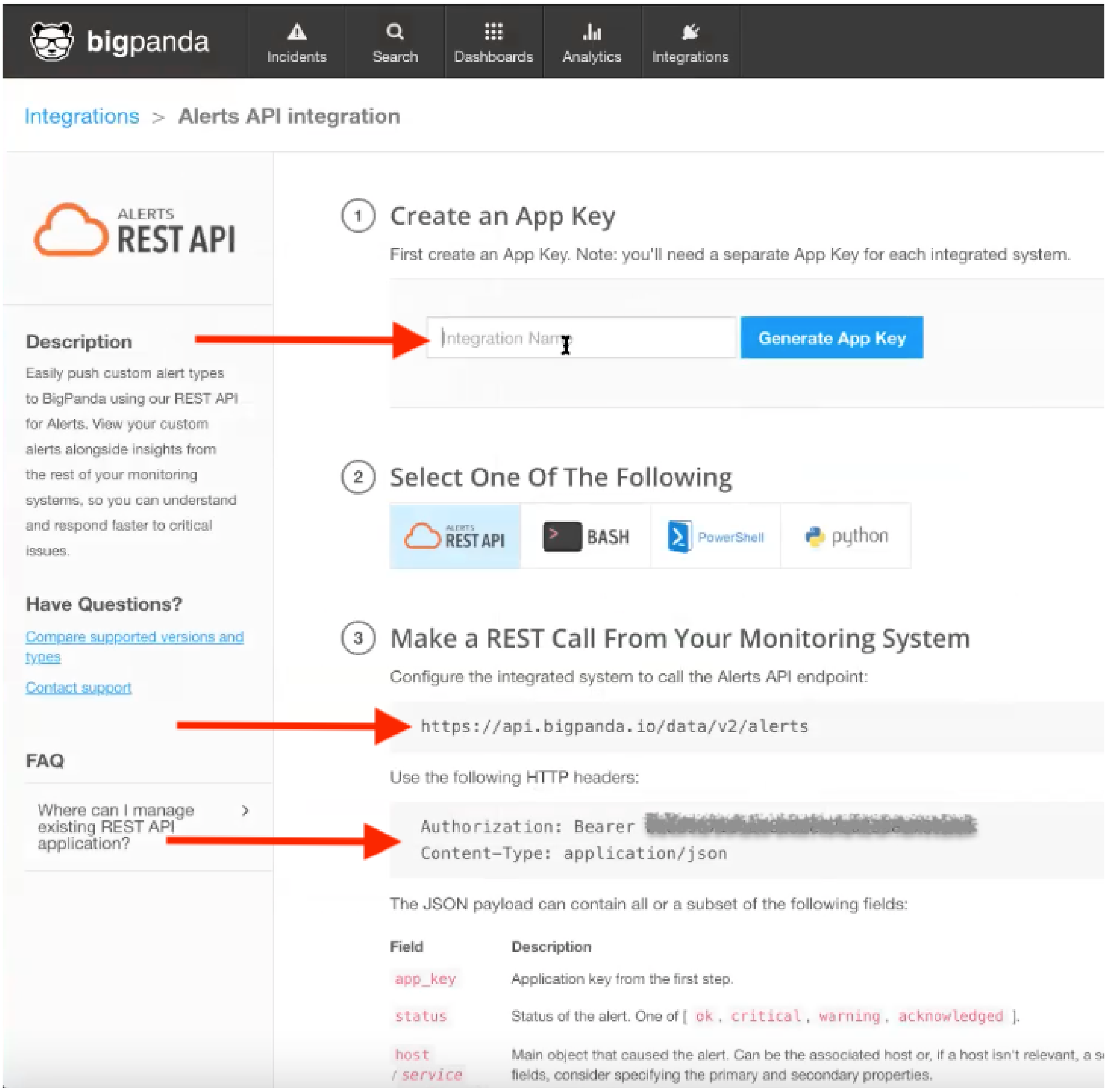
The **Integrations** tab

3. In the **Create a New Integration** modal, select **Alerts REST API** and click **Integrate**.



The **Create a New Integration** modal

- This opens the **Alerts API Integration** page.



Description

Easily push custom alert types to BigPanda using our REST API for Alerts. View your custom alerts alongside insights from the rest of your monitoring systems, so you can understand and respond faster to critical issues.

Have Questions?

Compare supported versions and types

Contact support

FAQ

Where can I manage existing REST API application?

① Create an App Key

First create an App Key. Note: you'll need a separate App Key for each integrated system.

② Select One Of The Following

③ Make a REST Call From Your Monitoring System

Configure the integrated system to call the Alerts API endpoint:

```
https://api.bigpanda.io/data/v2/alerts
```

Use the following HTTP headers:

```
Authorization: Bearer [redacted]
Content-Type: application/json
```

The JSON payload can contain all or a subset of the following fields:

Field	Description
`app_key`	Application key from the first step.
`status`	Status of the alert. One of [`ok`, `critical`, `warning`, `acknowledged`].
`host` / `service`	Main object that caused the alert. Can be the associated host or, if a host isn't relevant, a s fields, consider specifying the primary and secondary properties.

The **Alerts API 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. Store the following information from the **Make a REST Call From Your Monitoring System** section:

   - The Alerts API endpoint URL.
   - The `Authorization` HTTP header. This should consist of the word `Bearer`, a space, and a 32-character string. The 32-character string will be your **Access Token**.
   - The `Content-Type` HTTP header. This should be `application/json`.

You'll need the URL and header values when configuring Cribl Stream in the next section.

6. View your completed `Cribl Stream` integration in the **Integrations** tab.

Configure the Webhook Destination in Cribl Stream

1. From the top nav of a Cribl Stream instance or Group, 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.

2. In the **Configure** > **General Settings** tab, enter or select the following values:

   - **URL**: Enter the Alerts API endpoint URL, for example: `https://api.bigpanda.io/data/v2/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 following test input, substituting your own App Key value for the `<your_app_key>` placeholder shown:

```
[
  {
      "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.

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!

;

11.2.3. 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:

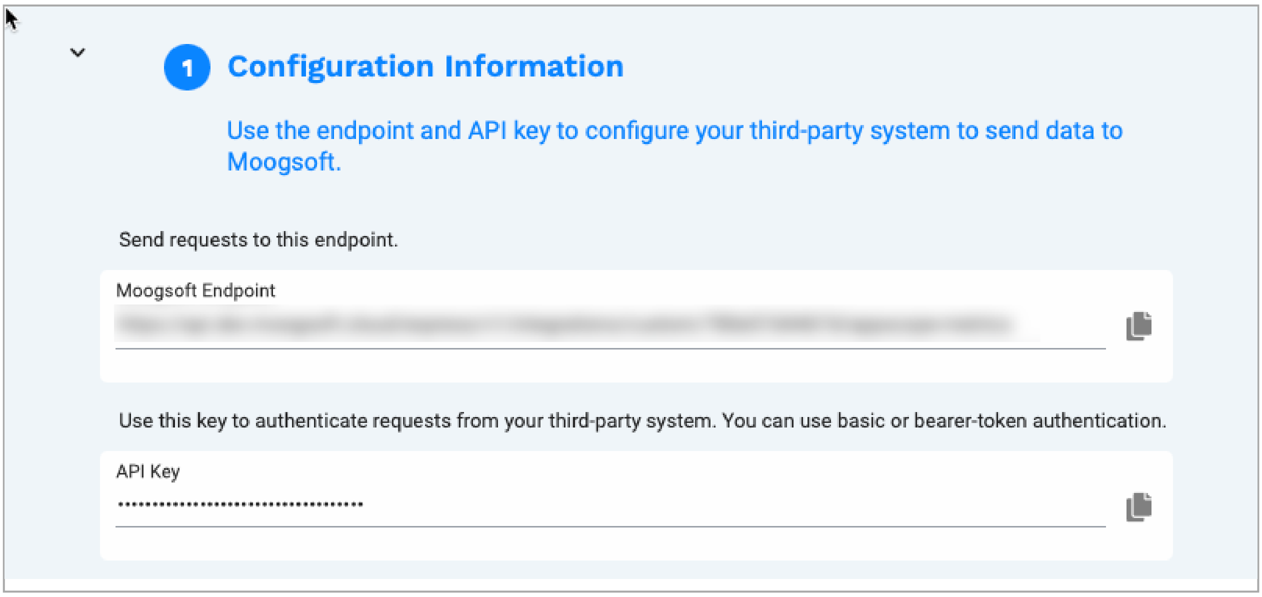
- A Moogsoft custom integration can ingest either events or metrics. Each custom integration has its own separate API key, and its own Cribl-to-Moogsoft mappings.
- There is no limit to the number of custom integrations you can create.
- The examples below illustrate simple event and metric mappings. 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 the top nav of a Cribl Stream instance or Group, 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.

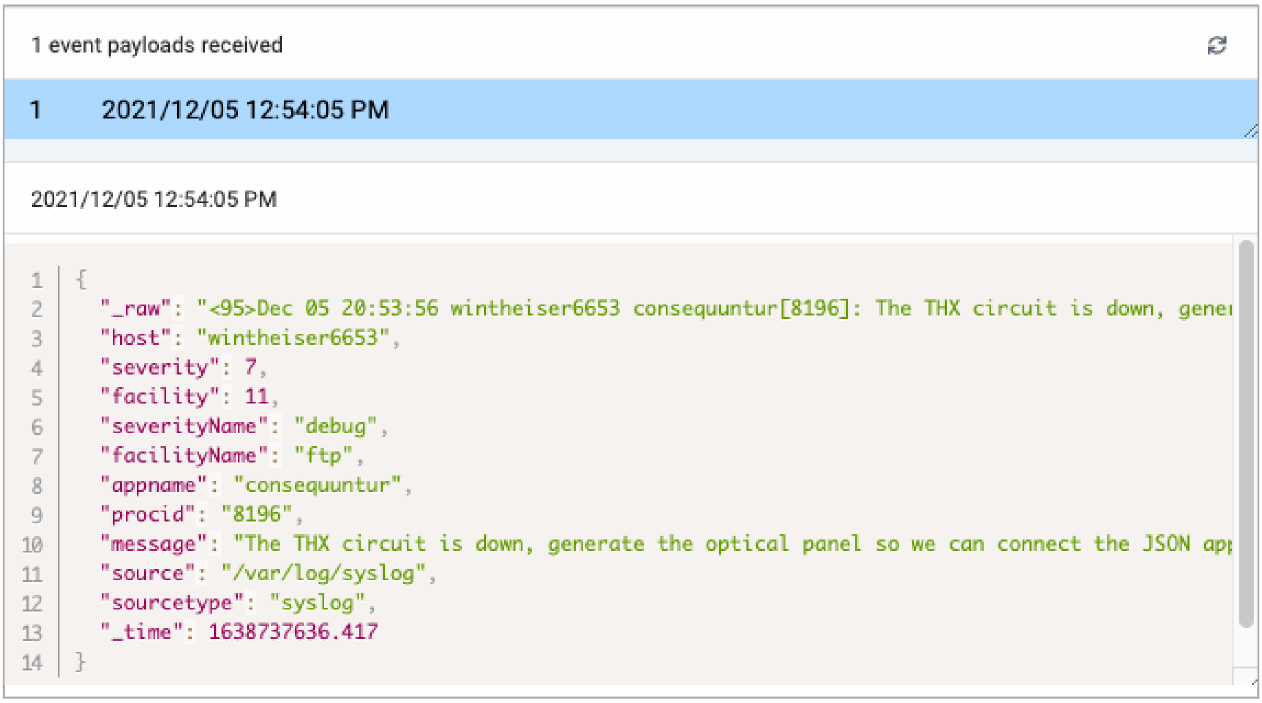
Send a Sample Payload to Moogsoft

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.



1 event payloads received

1 2021/12/05 12:54:05 PM

2021/12/05 12:54:05 PM

```
{
  "_raw": "<95>Dec 05 20:53:56 wintheiser6653 consequuntur[8196]: The THX circuit is down, gene
  "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 app
  "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

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.

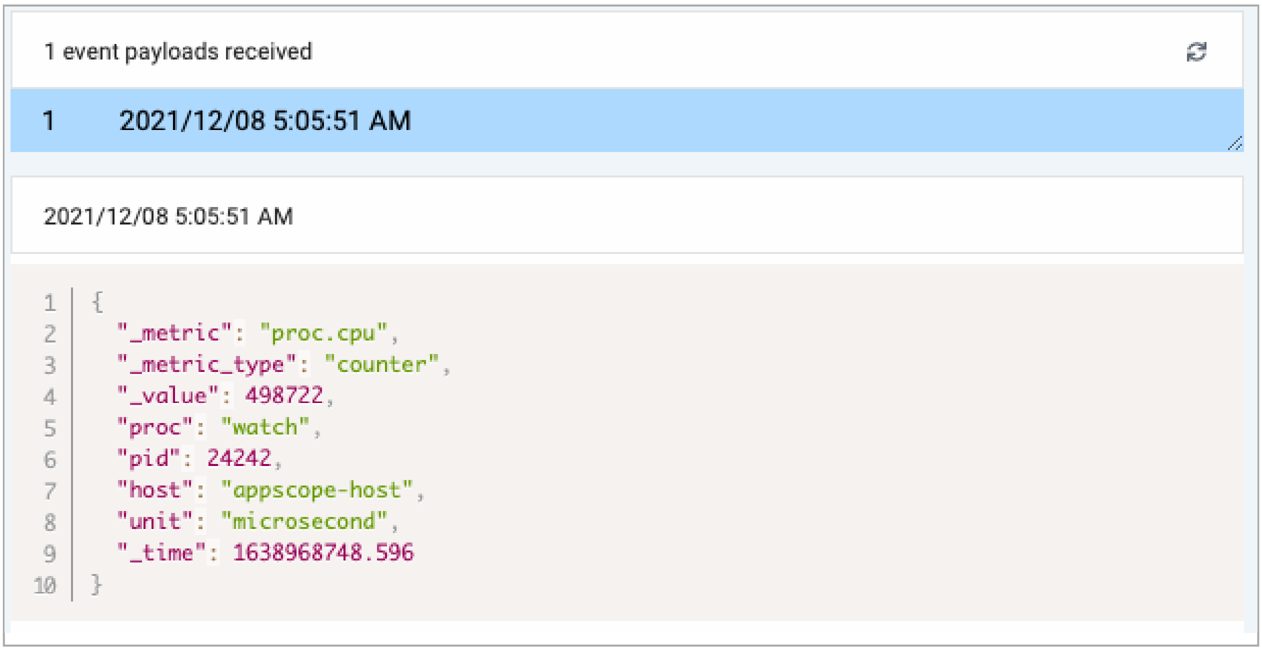
Send a Sample Payload to Moogsoft

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.



1 event payloads received

1 2021/12/08 5:05:51 AM

2021/12/08 5:05:51 AM

```
{
  "_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`
`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.

;

11.2.4. 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.

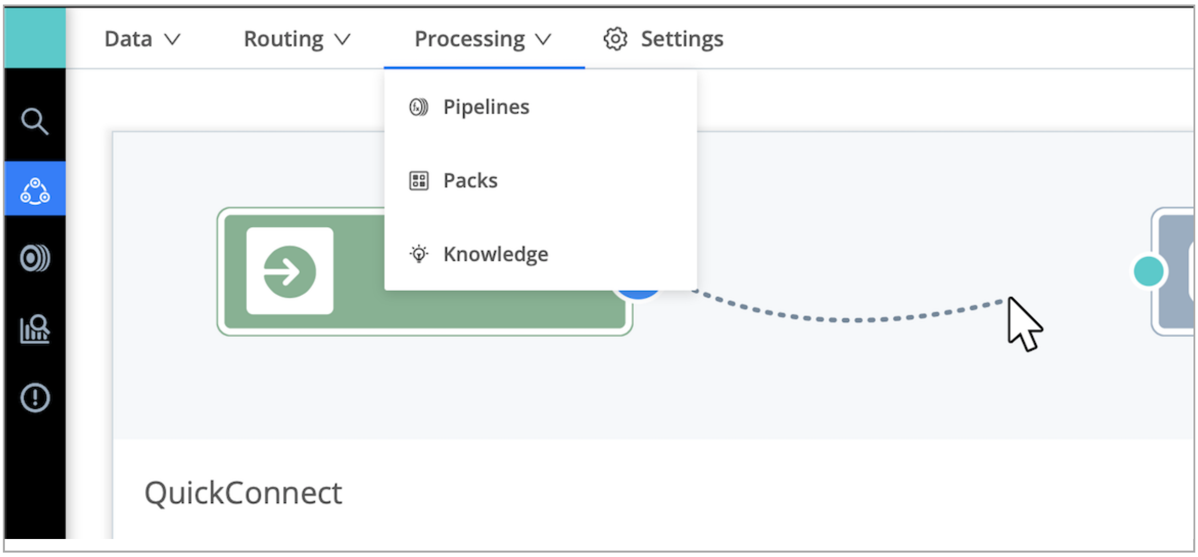
4. Create the Detection Rules and/or Policy that you want Nightfall to scan with.

   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.

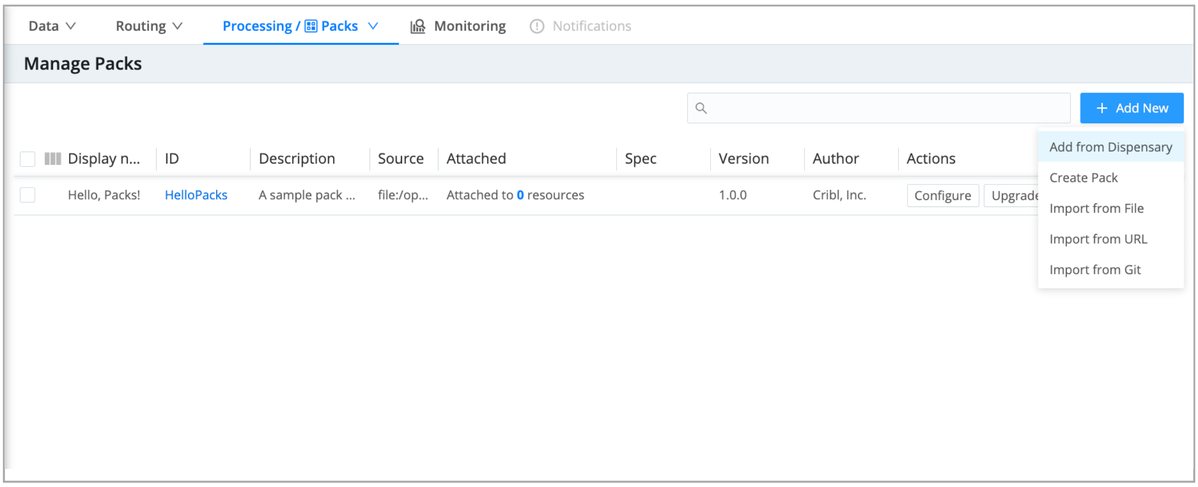
Installing the Nightfall Pack

1. From Cribl Stream's top nav, select **Processing** > **Packs**.



Selecting Packs

2. Click **+ Add New**, then select **Add from Dispensary** to open the Packs Dispensary drawer.

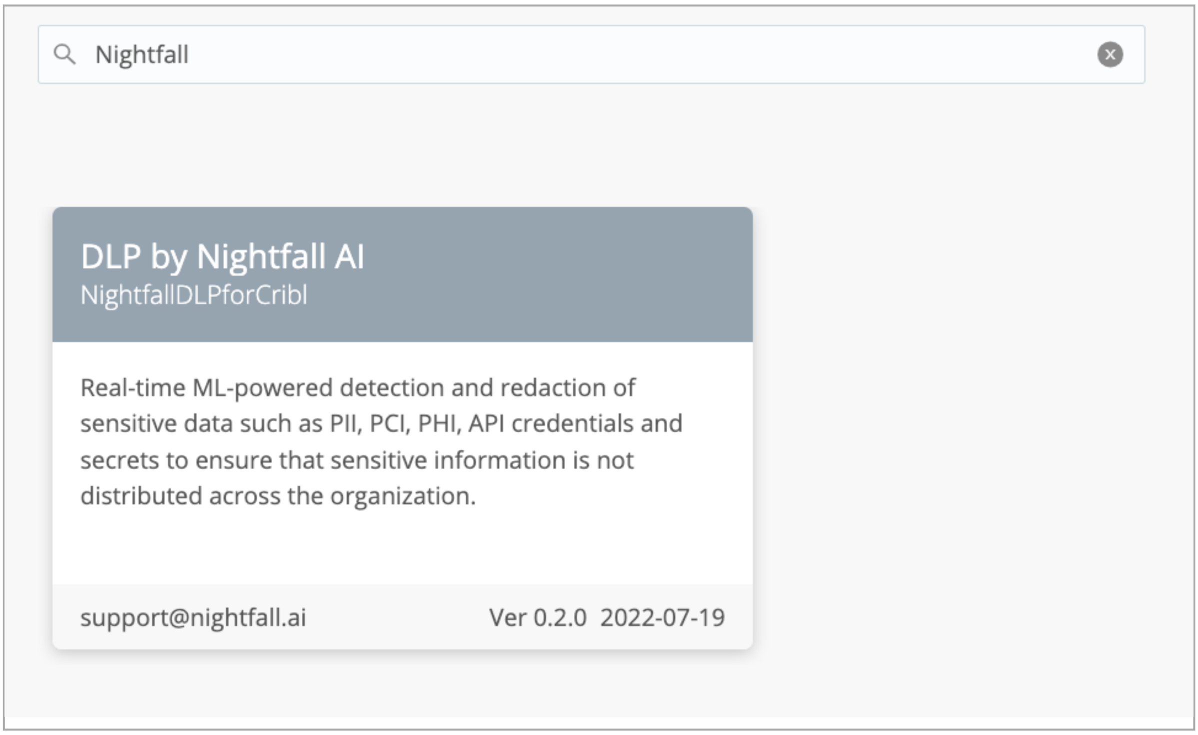


Manage Packs

Display n...	ID	Description	Source	Attached	Spec	Version	Author	Actions
Hello, Packs!	HelloPacks	A sample pack ...	file:/op...	Attached to 0 resources		1.0.0	Cribl, Inc.	Configure Upgrade

Adding a new Pack from the Dispensary

3. Navigate to the **DLP by Nighfall AI** Pack's tile.



Nightfall

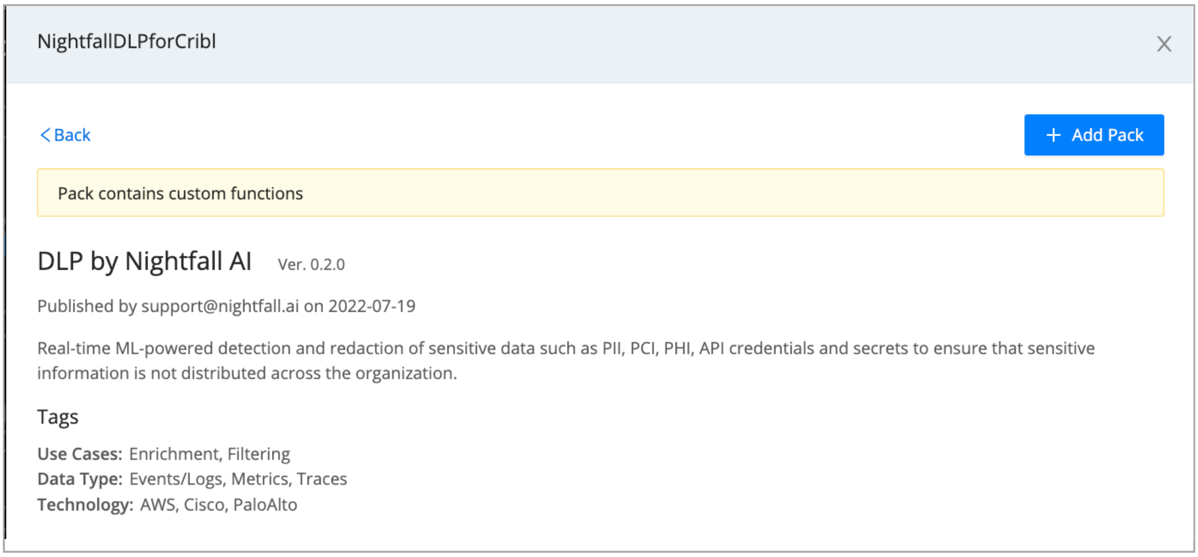
DLP by Nightfall AI
NightfallDLPforCribl

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

4. 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

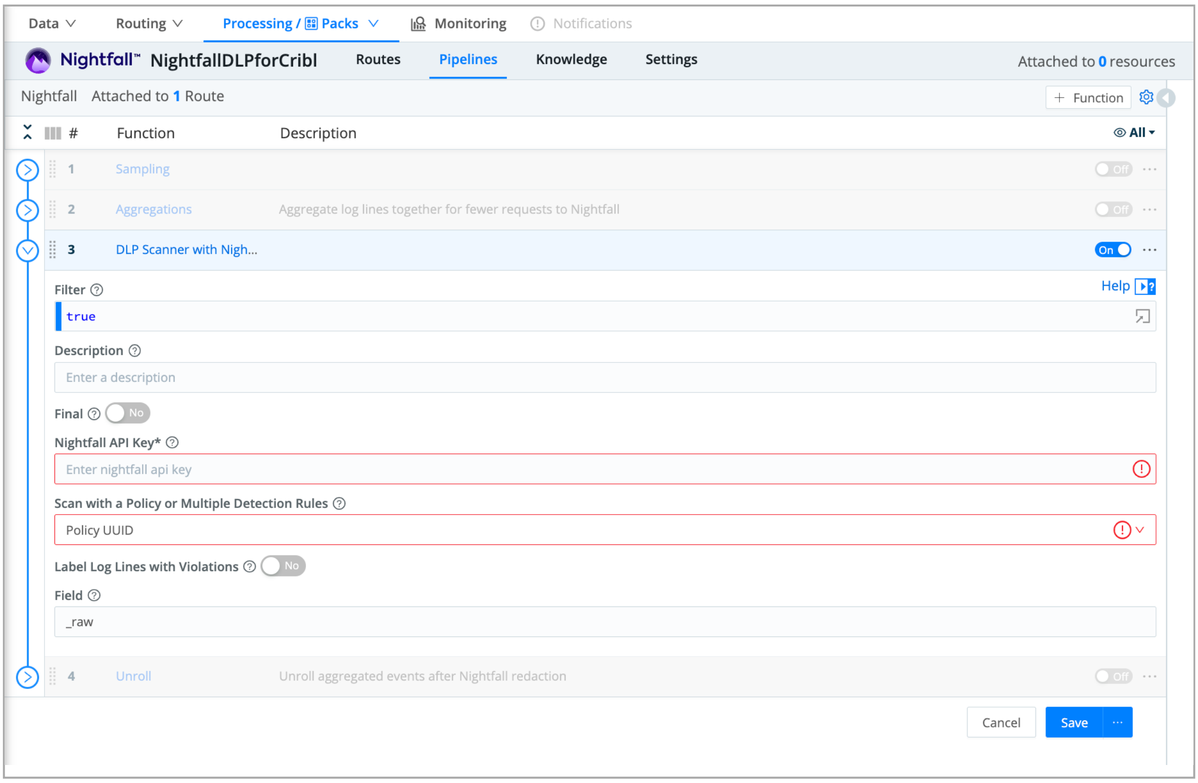
5. Click **+ Add Pack**. Shortly, you'll see a banner confirming that the Pack is installed.

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.



#	Function	Description	
1	Sampling		Off
2	Aggregations	Aggregate log lines together for fewer requests to Nightfall	Off
3	DLP Scanner with Nigh...		On
4	Unroll	Unroll aggregated events after Nightfall redaction	Off

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

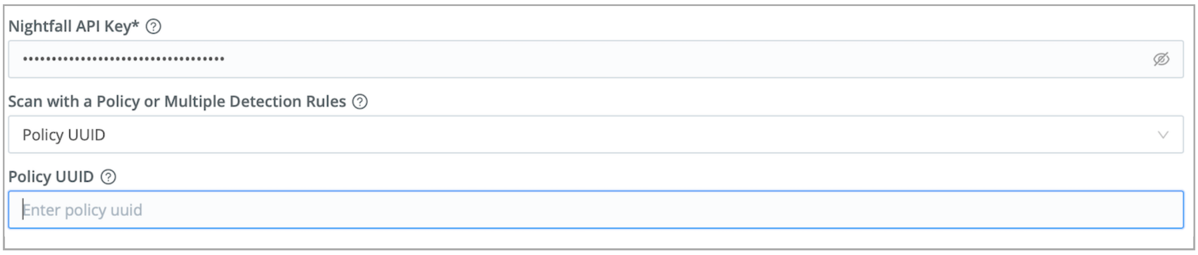
Selecting the Nightfall Pipeline

4. In the **Nightfall API Key** field, enter the API key you created in the Nightfall Dashboard.

5. 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.



Nightfall API Key*: ••••••••••••••••••••••••••••••••••••

Scan with a Policy or Multiple Detection Rules: Policy UUID

Policy UUID: Enter policy uuid

Scanning with a Policy

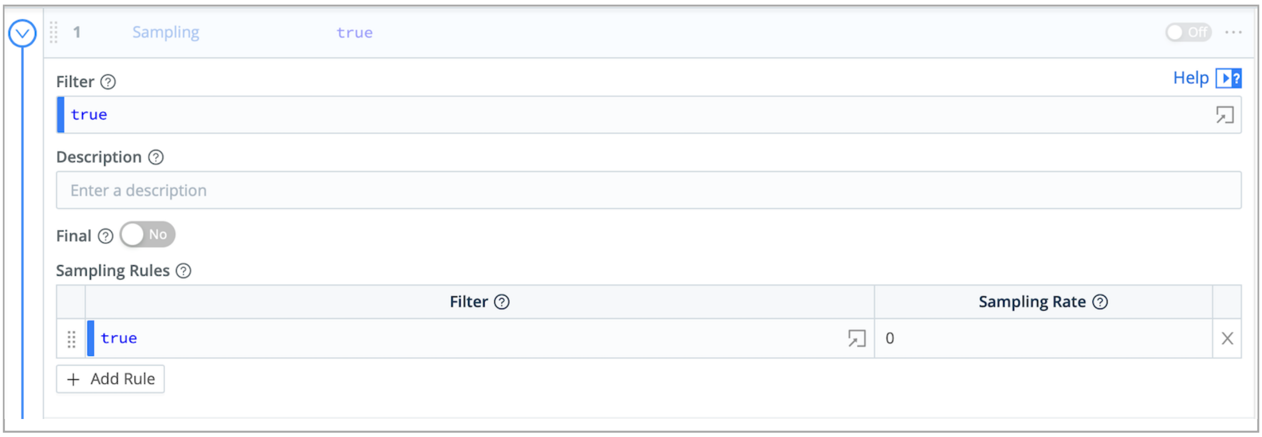
6. (Optionally:) Label sensitive events. If you toggle **Label Log Lines with nightfall_violations** to `Yes`, each event where Nightfall has detected sensitive information will have an added `nightfall_violations: true` field.

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.



1 Sampling true Off

Filter: true

Description: Enter a description

Final: No

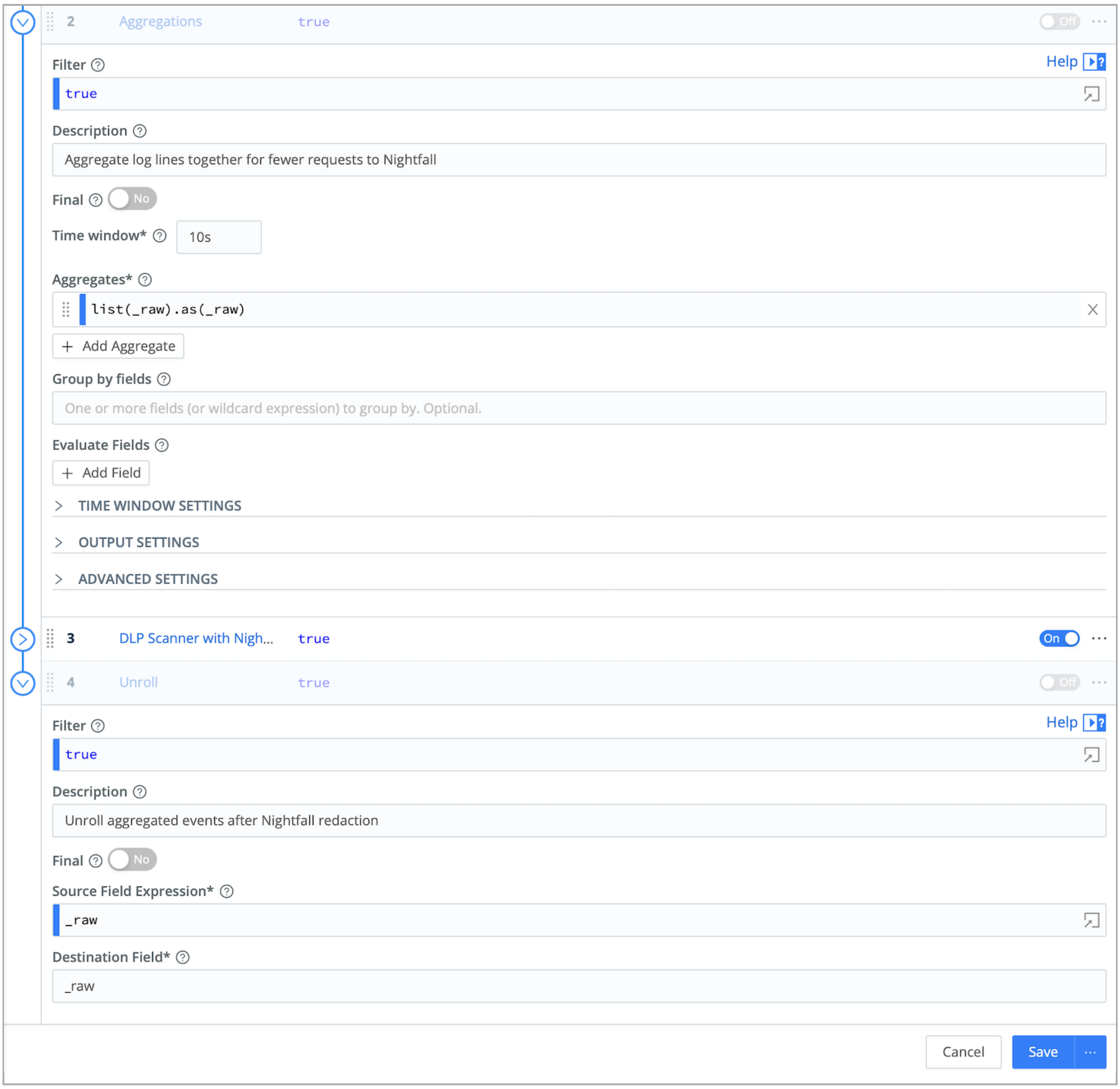
Sampling Rules

Filter	Sampling Rate
true	0

Disabled steps

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**.



2 Aggregations true

Filter: true

Description: Aggregate log lines together for fewer requests to Nightfall

Final: No

Time window*: 10s

Aggregates*: list(_raw).as(_raw)

Group by fields: One or more fields (or wildcard expression) to group by. Optional.

Evaluate Fields

3 DLP Scanner with Nigh... true On

4 Unroll true

Filter: true

Description: Unroll aggregated events after Nightfall redaction

Final: No

Source Field Expression*: _raw

Destination Field*: _raw

Enabling log batching

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.

;

11.2.5. 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.

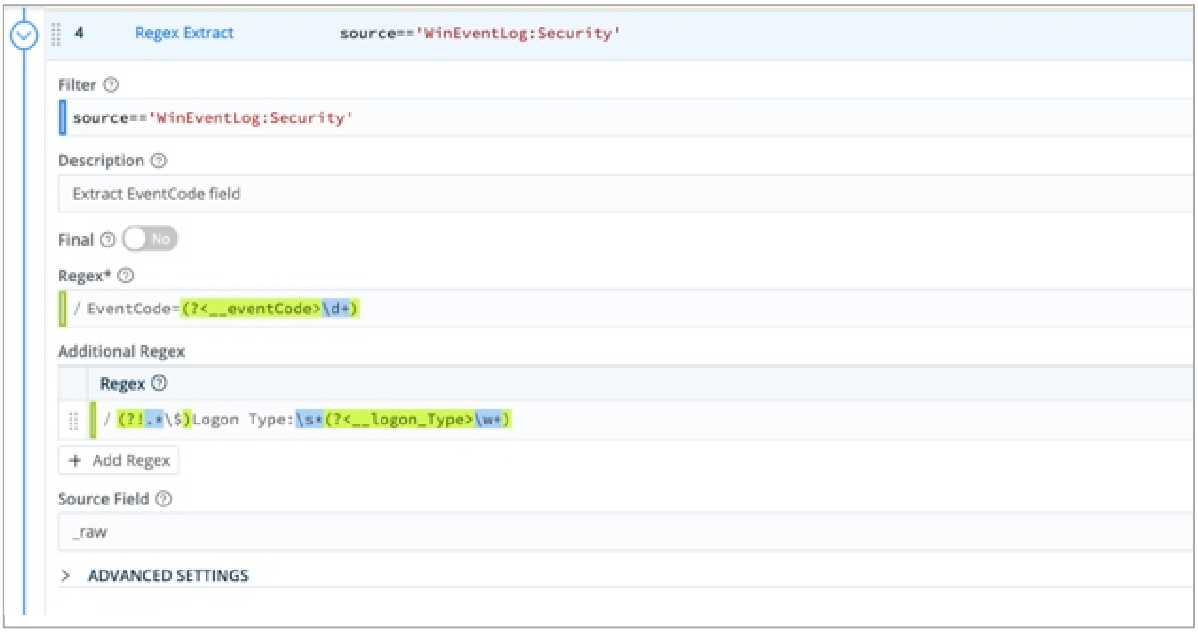
Dropping the Excess Baggage

> 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.



4 Regex Extract `source=='WinEventLog:Security'`

Filter: `source=='WinEventLog:Security'`

Description: Extract EventCode field

Final: No

Regex*: `/ EventCode=(?<__eventCode>\d+)`

Additional Regex

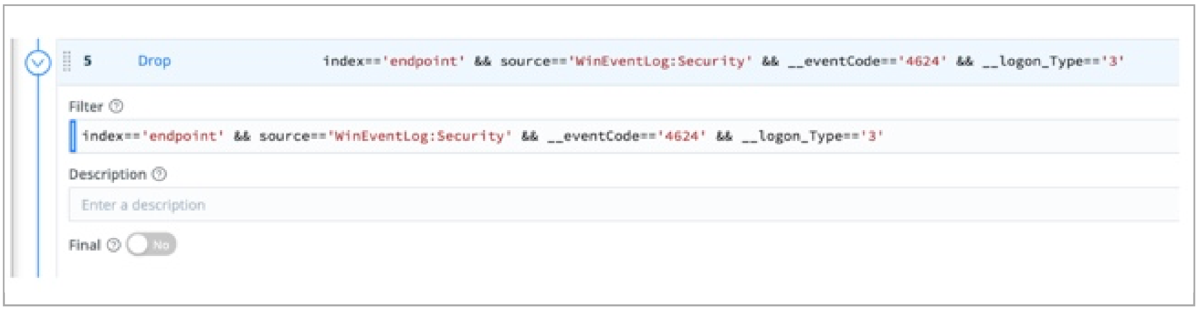
Regex: `/ (?!.*\$)Logon Type:\s*(?<__logon_Type>\w+)`

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' &&
__logon_Type=='3'
```



5 Drop `index=='endpoint' && source=='WinEventLog:Security' && __eventCode=='4624' && __logon_Type=='3'`

Filter: `index=='endpoint' && source=='WinEventLog:Security' && __eventCode=='4624' && __logon_Type=='3'`

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

;

11.2.6. Splunk to Elasticsearch

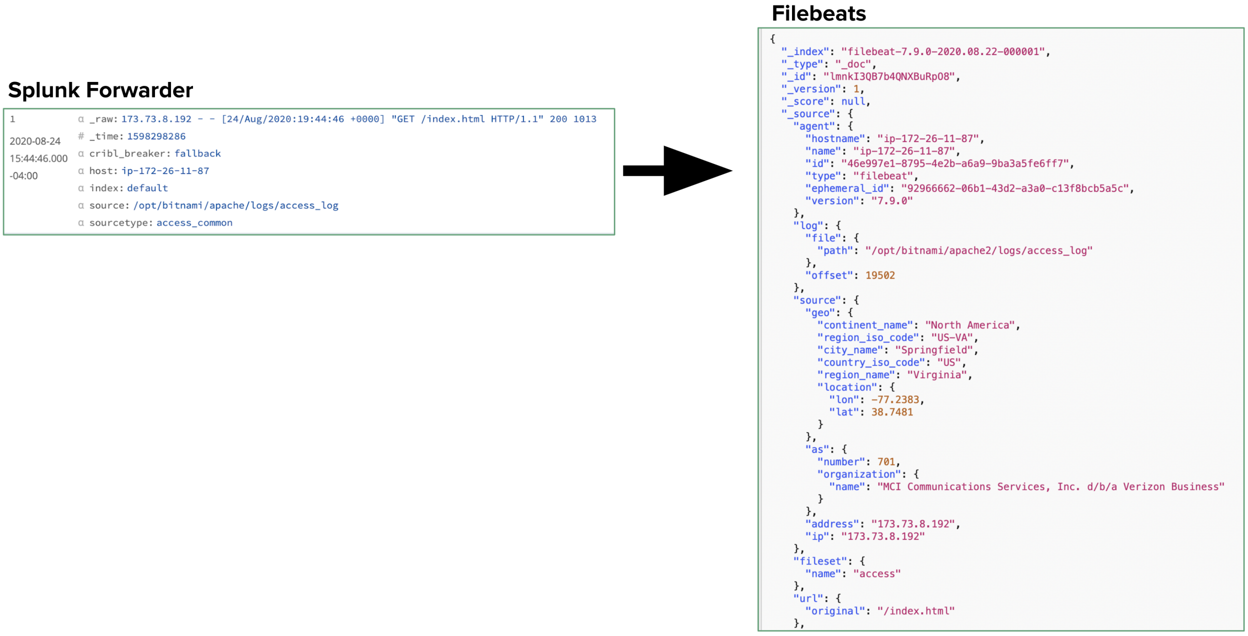
To route data from existing Splunk infrastructure to Elasticsearch services, you might face a daunting task: re-architecting your entire forwarding tier. This could require retooling lots of servers – up to hundreds, or thousands – to uninstall their Splunk forwarders, and swap in Elastic-compatible 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.



Splunk Forwarder

```
1                 α _raw: 173.73.8.192 - - [24/Aug/2020:19:44:46 +0000] "GET /index.html HTTP/1.1" 200 1013
                  # _time: 1598298286
2020-08-24        α cribl_breaker: fallback
15:44:46.000      α host: ip-172-26-11-87
-04:00            α index: default
                  α source: /opt/bitnami/apache/logs/access_log
                  α sourcetype: access_common
```

Filebeats

```
{
  "_index": "filebeat-7.9.0-2020.08.22-000001",
  "_type": "_doc",
  "_id": "lmnkI3QB7b4QNXBuRp08",
  "_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"
    },
```

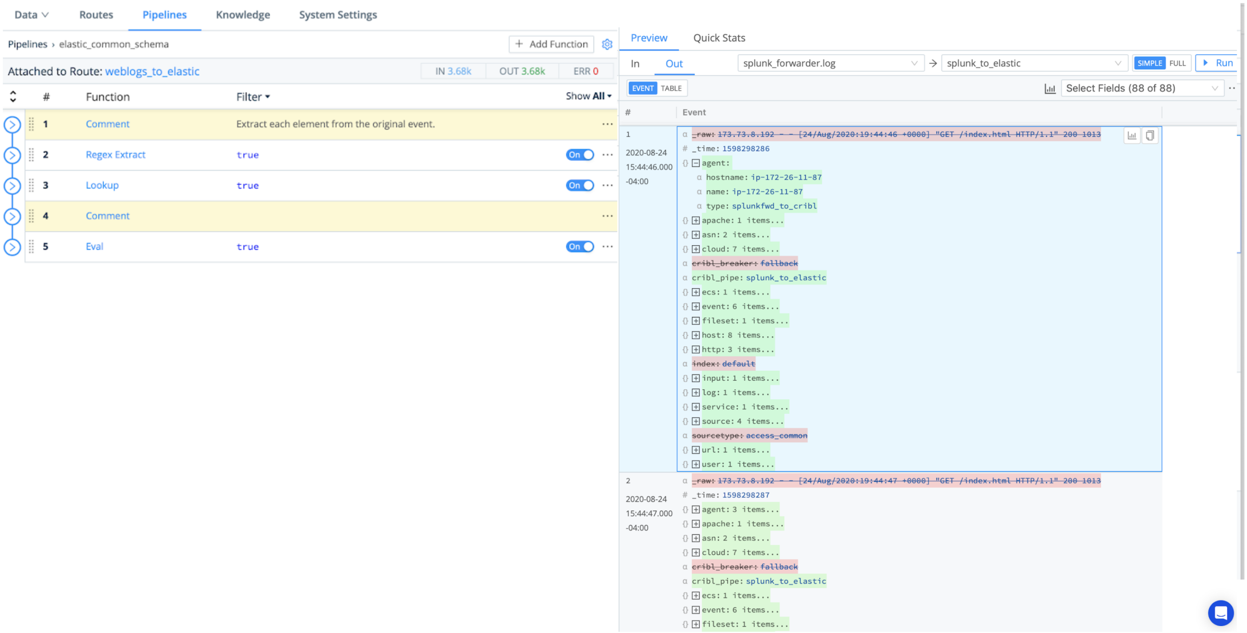
Transforming to Elastic Common Schema

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.



#	Function	Filter	
1	Comment	Extract each element from the original event.	
2	Regex Extract	true	On
3	Lookup	true	On
4	Comment		
5	Eval	true	On

```
#   Event
1   ~~_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   agent:
-04:00       hostname: ip-172-26-11-87
             name: ip-172-26-11-87
             type: splunkfwd_to_cribl
    apache: 1 items...
    asn: 2 items...
    cloud: 7 items...
    ~~cribl_breaker: fallback~~
    cribl_pipe: splunk_to_elastic
    ecs: 1 items...
    event: 6 items...
    fileset: 1 items...
    host: 8 items...
    http: 3 items...
    ~~index: default~~
    input: 1 items...
    log: 1 items...
    service: 1 items...
    source: 4 items...
    ~~sourcetype: access_common~~
    url: 1 items...
    user: 1 items...
2   ~~_raw: 173.73.8.192 - - [24/Aug/2020:19:44:47 +0000] "GET /index.html HTTP/1.1" 200 1013~~
2020-08-24   _time: 1598298287
15:44:47.000   agent: 3 items...
-04:00   apache: 1 items...
    asn: 2 items...
    cloud: 7 items...
    ~~cribl_breaker: fallback~~
    cribl_pipe: splunk_to_elastic
    ecs: 1 items...
    event: 6 items...
    fileset: 1 items...
```

Cribl Stream Pipeline and Data Preview

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

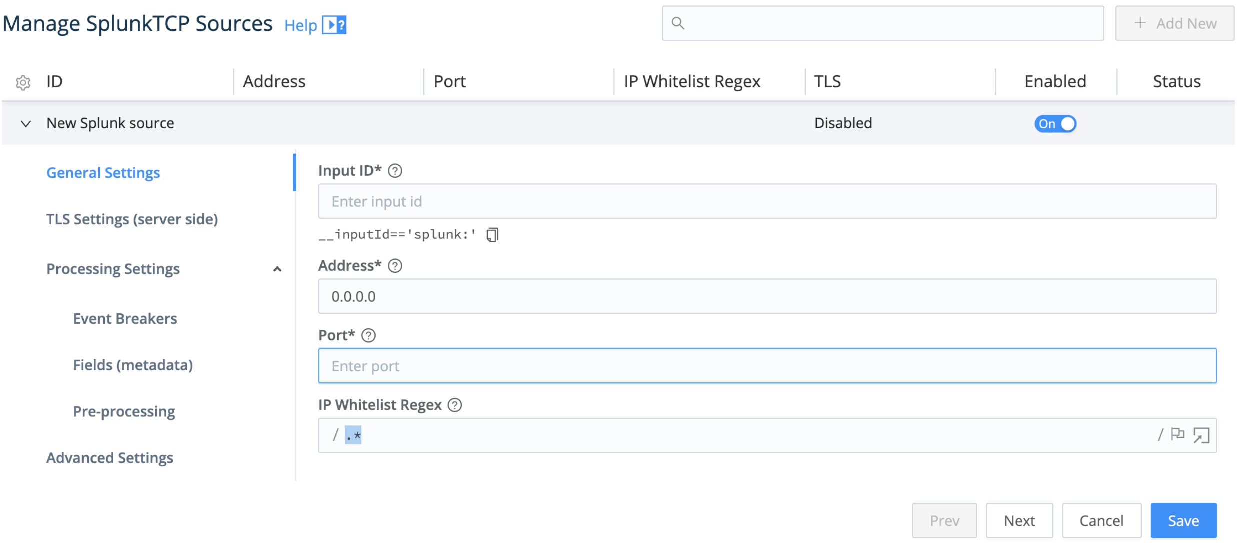
First, in a Splunk App, configure a Splunk forwarder (UF or HF) to specify your Cribl Workers as destinations. Use outputs.conf stanzas of this form:

```
[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.)



ID	Address	Port	IP Whitelist Regex	TLS	Enabled	Status
New Splunk source				Disabled	On	

Input ID*: Enter input id

`__inputId=='splunk:'`

Address*: 0.0.0.0

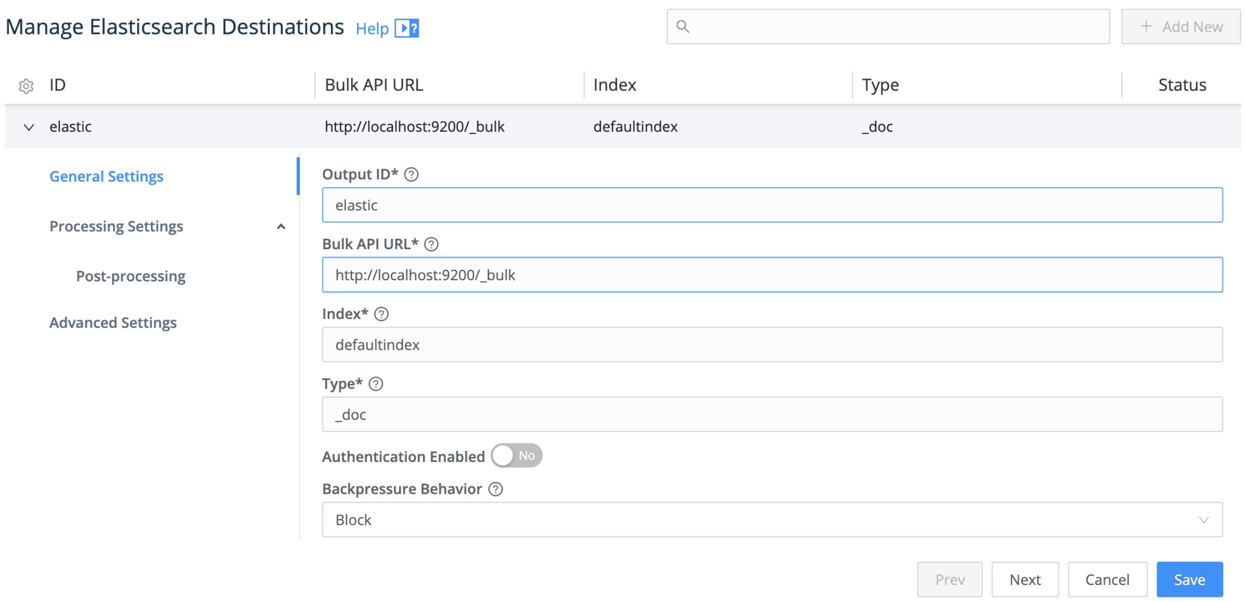
Port*: Enter port

IP Whitelist Regex: / .* /

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**.



Manage Elasticsearch Destinations

ID	Bulk API URL	Index	Type	Status
elastic	http://localhost:9200/_bulk	defaultindex	_doc	

Output ID*: elastic

Bulk API URL*: http://localhost:9200/_bulk

Index*: defaultindex

Type*: _doc

Authentication Enabled: No

Backpressure Behavior: Block

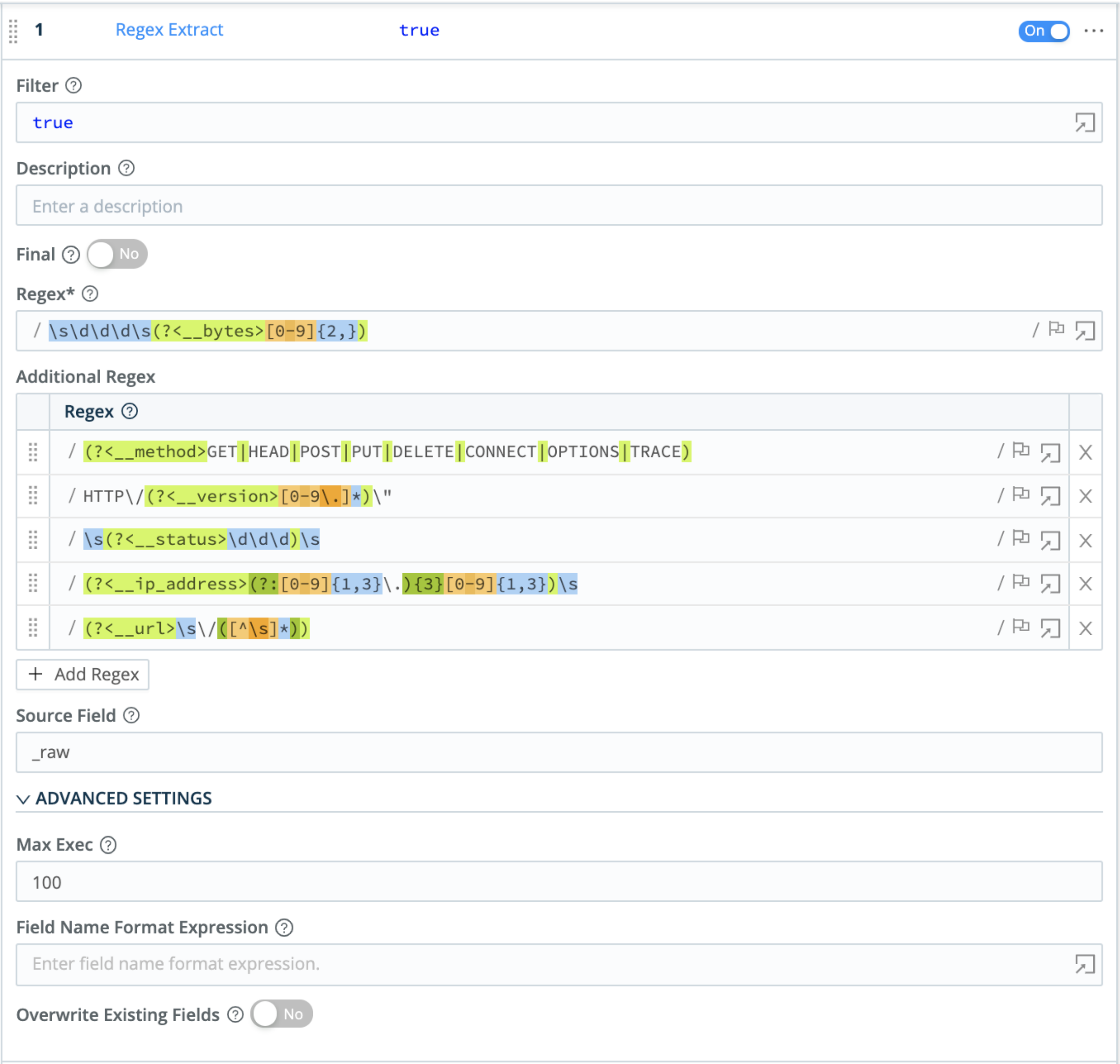
Elasticsearch Destination configuration

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:



1 Regex Extract `true` On

Filter: `true`

Description: Enter a description

Final: No

Regex*: `/ \s\d\d\d\s(?<__bytes>[0-9]{2,}) /`

Additional 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

ADVANCED SETTINGS

Max Exec: 100

Field Name Format Expression: Enter field name format expression.

Overwrite Existing Fields: No

Regex Extract Function

Here are the six rows of regex in this example:

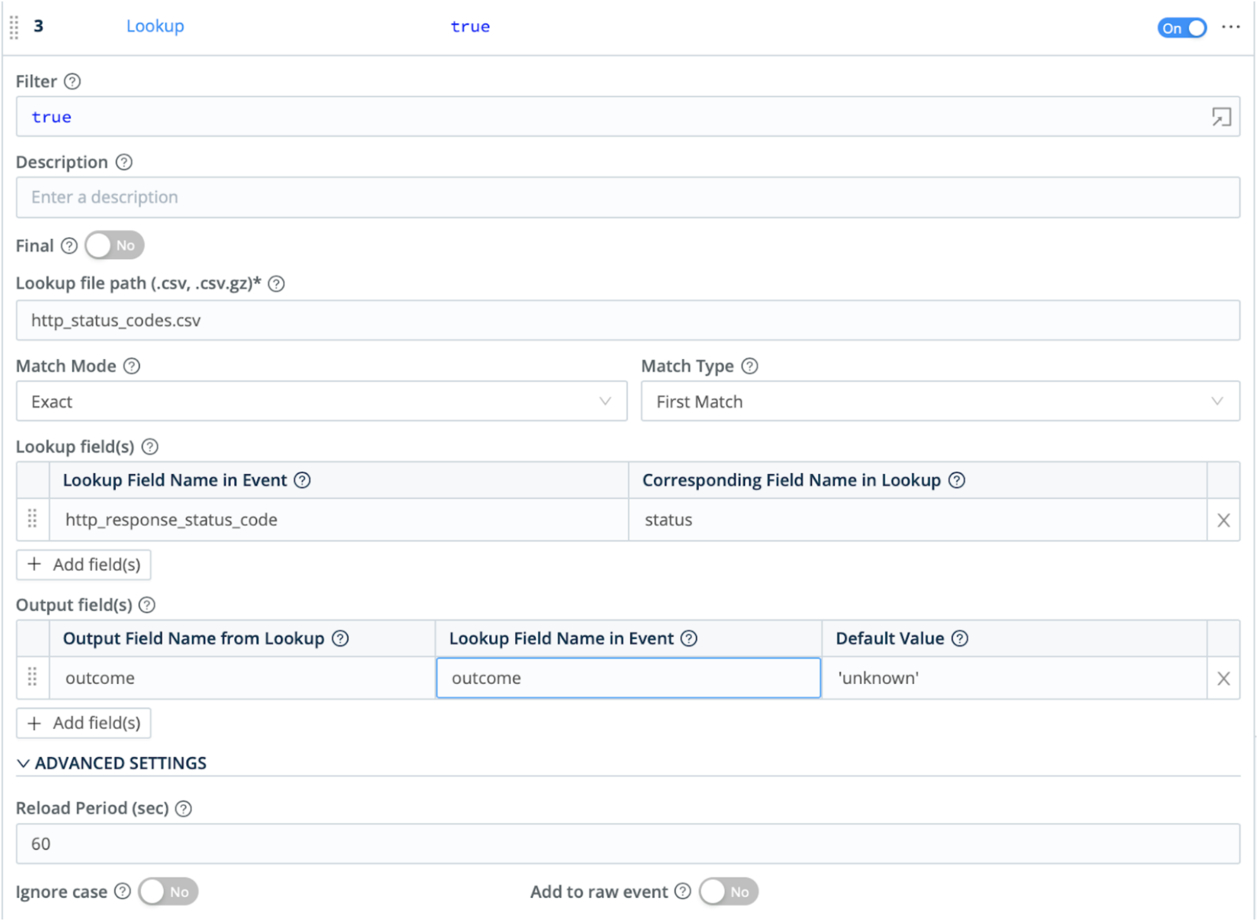
Regex; Additional Regex

```
/\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.



3 Lookup `true` On

Filter: `true`

Description: Enter a description

Final: No

Lookup file path (.csv, .csv.gz)*: http_status_codes.csv

Match Mode: Exact

Match Type: First Match

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
http_response_status_code	status

+ Add field(s)

Output field(s)

Output Field Name from Lookup	Lookup Field Name in Event	Default Value
outcome	outcome	'unknown'

+ Add field(s)

ADVANCED SETTINGS

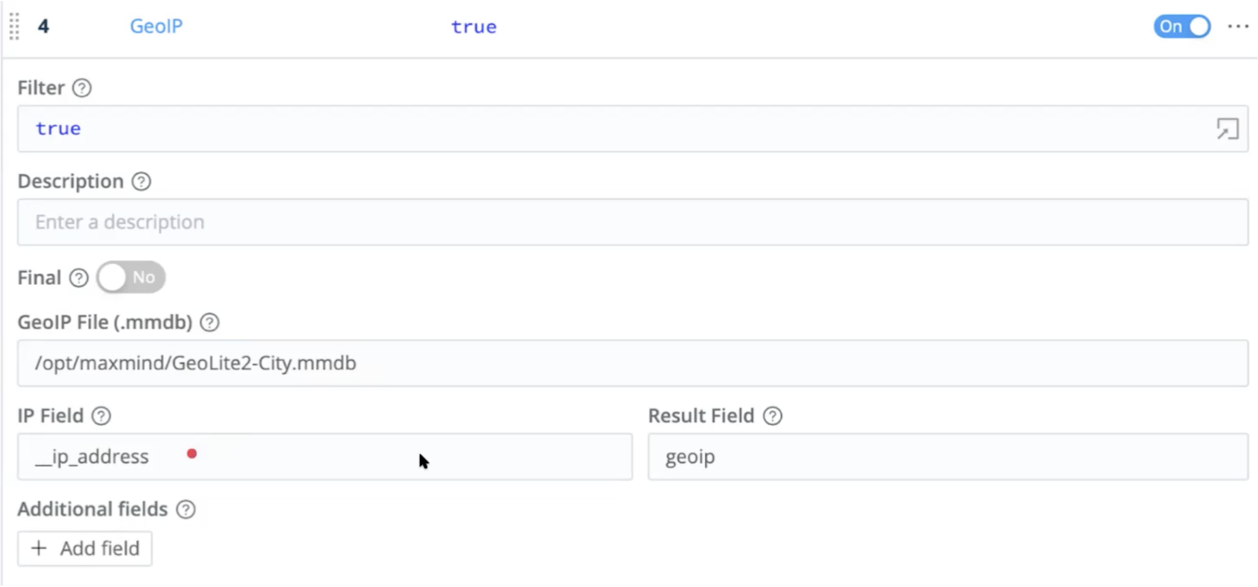
Reload Period (sec): 60

Ignore case: No

Add to raw event: No

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`.



4 GeoIP true

Filter: true

Description: Enter a description

Final: No

GeoIP File (.mmdb): /opt/maxmind/GeoLite2-City.mmdb

IP Field: __ip_address

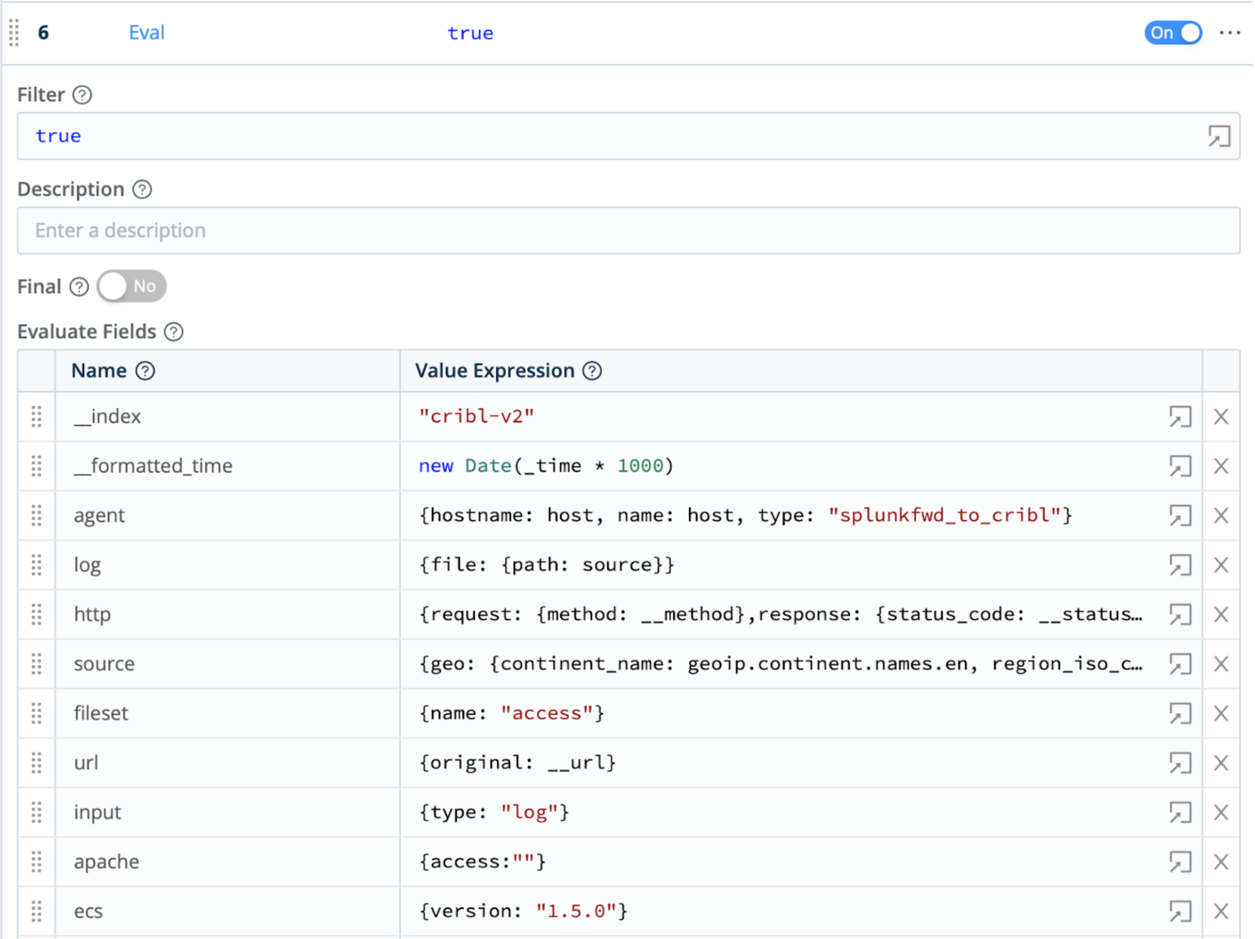
Result Field: geoip

Additional fields

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.



6 Eval true

Filter: true

Description: Enter a description

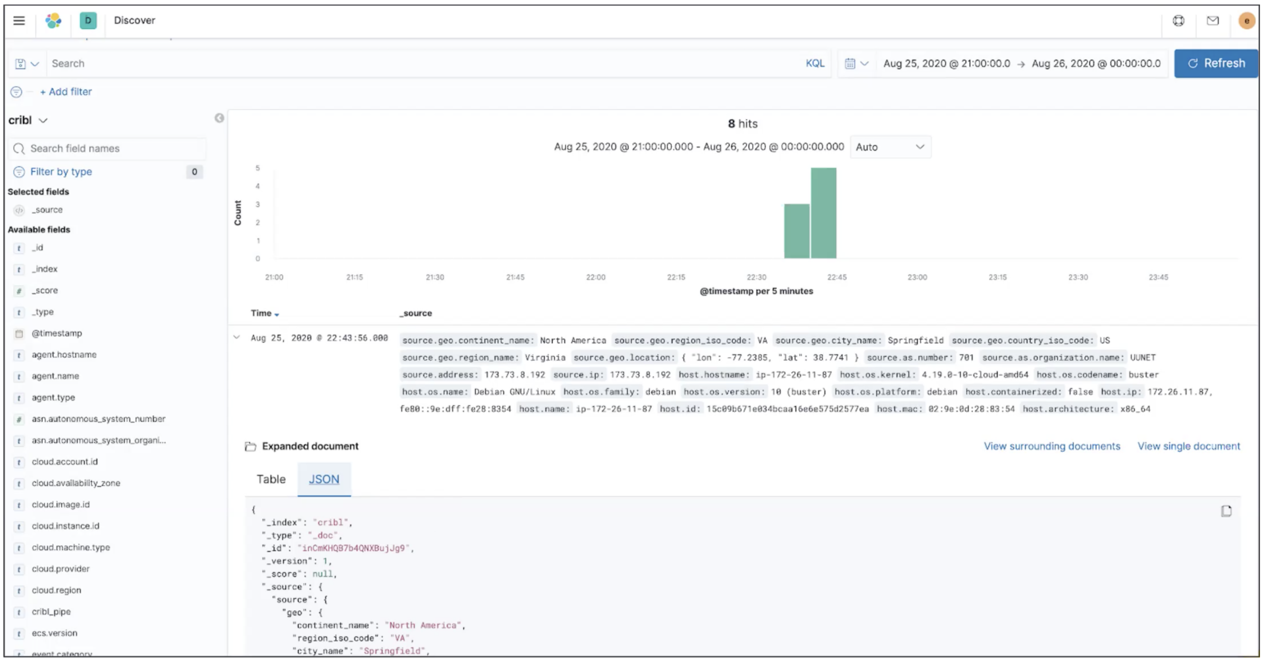
Final: No

Evaluate Fields

Name	Value Expression
__index	"cribl-v2"
__formatted_time	new Date(_time * 1000)
agent	{hostname: host, name: host, type: "splunkfwd_to_cribl"}
log	{file: {path: source}}
http	{request: {method: __method},response: {status_code: __status…
source	{geo: {continent_name: geoip.continent.names.en, region_iso_c…
fileset	{name: "access"}
url	{original: __url}
input	{type: "log"}
apache	{access:""}
ecs	{version: "1.5.0"}

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

;

11.2.7. 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.

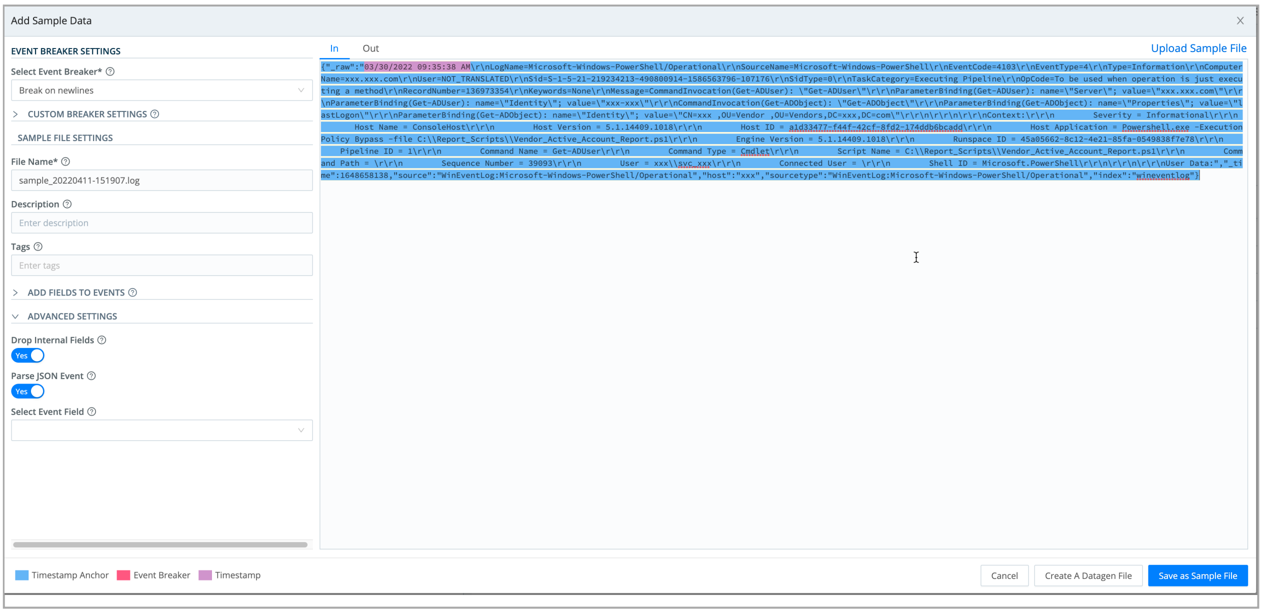
In this guide, we'll show you how to:

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.



Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*: Break on newlines

CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*: sample_20220411-151907.log

Description: Enter description

Tags: Enter tags

ADD FIELDS TO EVENTS

ADVANCED SETTINGS

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"}
```

Saving captured data

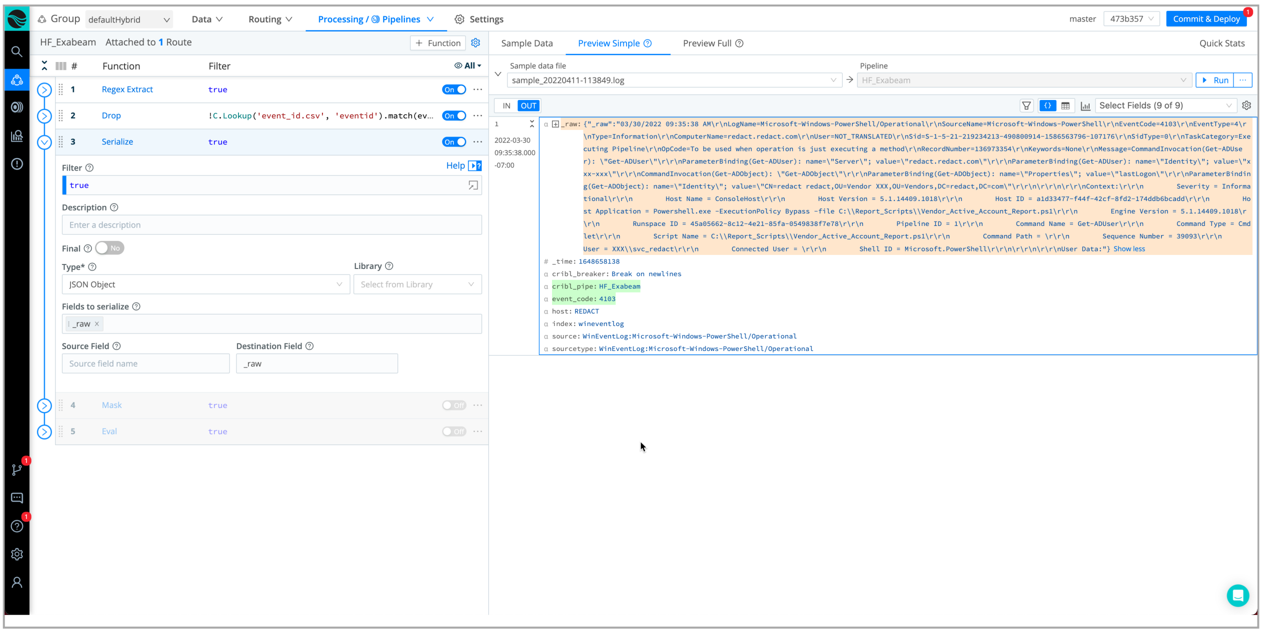
In your production environment, you can filter incoming events in real time (e.g., using a filter expression like `Windows Security logging`) to identify your in-scope Exabeam data.

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.



#	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

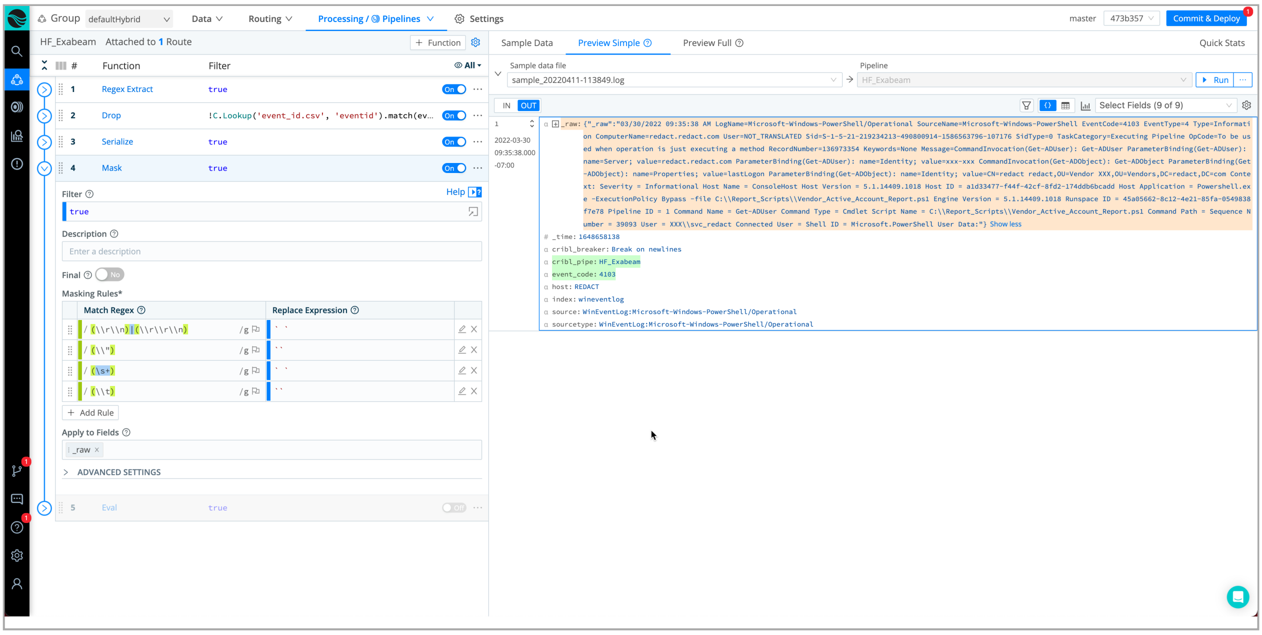
```
1  2022-03-30 09:35:38.000 -07:00
_raw: {"_raw":"03/30/2022 09:35:38 AM\r\nLogName=Microsoft-Windows-PowerShell/Operational\r\nSourceName=Microsoft-Windows-PowerShell\r\nEventCode=4103\r\nEventType=4\r\nType=Information\r\nComputerName=redact.redact.com\r\nUser=NOT_TRANSLATED\r\nSid=S-1-5-21-219234213-490800914-1586563796-107176\r\nSidType=0\r\nTaskCategory=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.



#	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

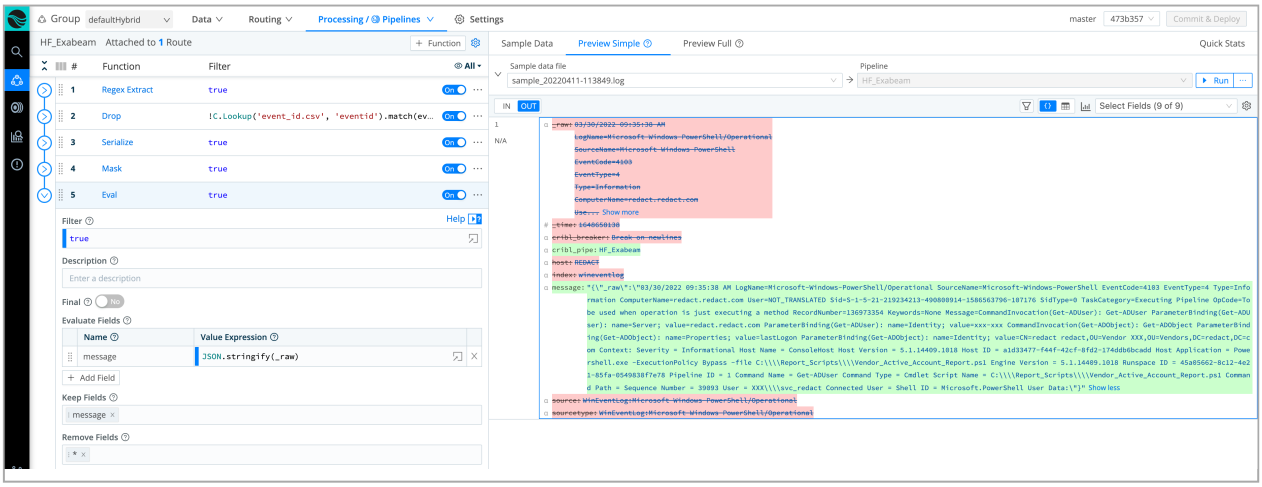
Sample data file: sample_20220411-113849.log
Pipeline: HF_Exabeam

```
1  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
2	Drop	!C.Lookup('event_id.csv', 'eventid').match(ev...
3	Serialize	true
4	Mask	true
5	Eval	true

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

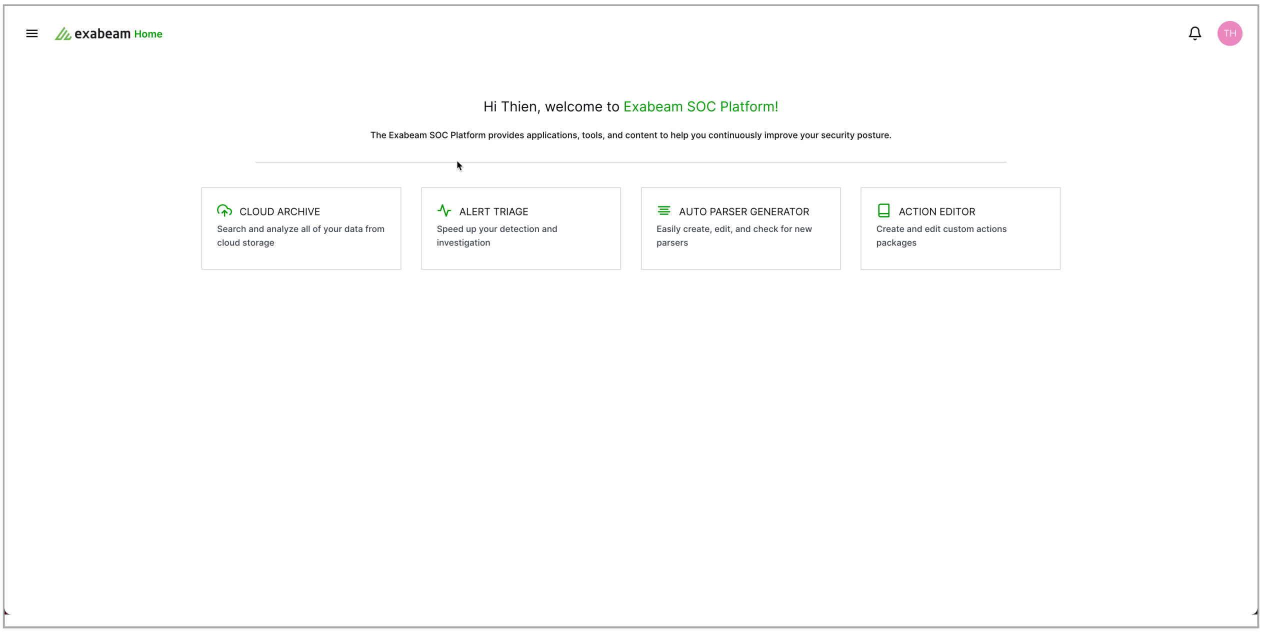
Select Fields (9 of 9)

```
1
N/A
~~_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~~
      ~~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=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
~~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

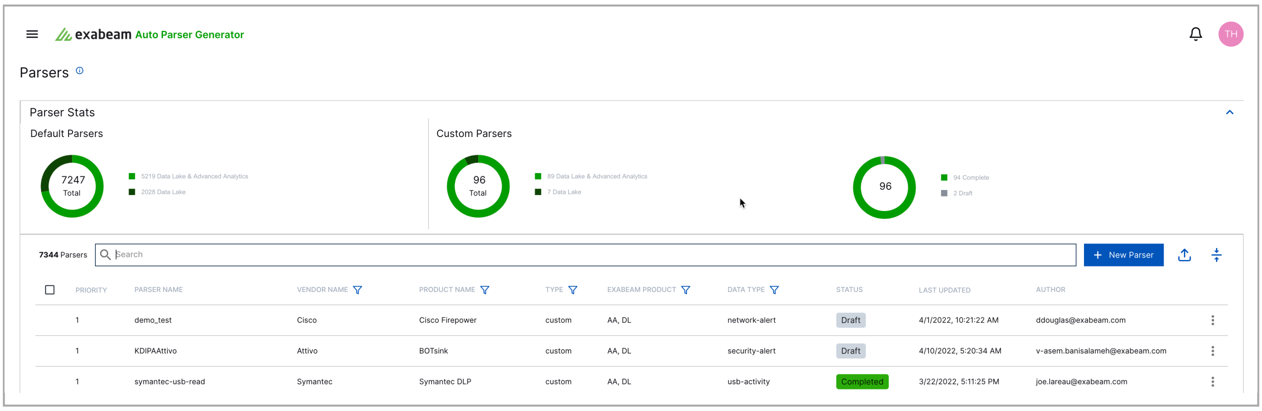
ALERT TRIAGE
Speed up your detection and investigation

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**.



exabeam Auto Parser Generator

Parsers

Parser Stats

Default Parsers

7247 Total

🟩 5219 Data Lake & Advanced Analytics
🟩 2028 Data Lake

Custom Parsers

96 Total

🟩 89 Data Lake & Advanced Analytics
🟩 7 Data Lake

96

🟩 94 Complete
▨ 2 Draft

7344 Parsers

Search

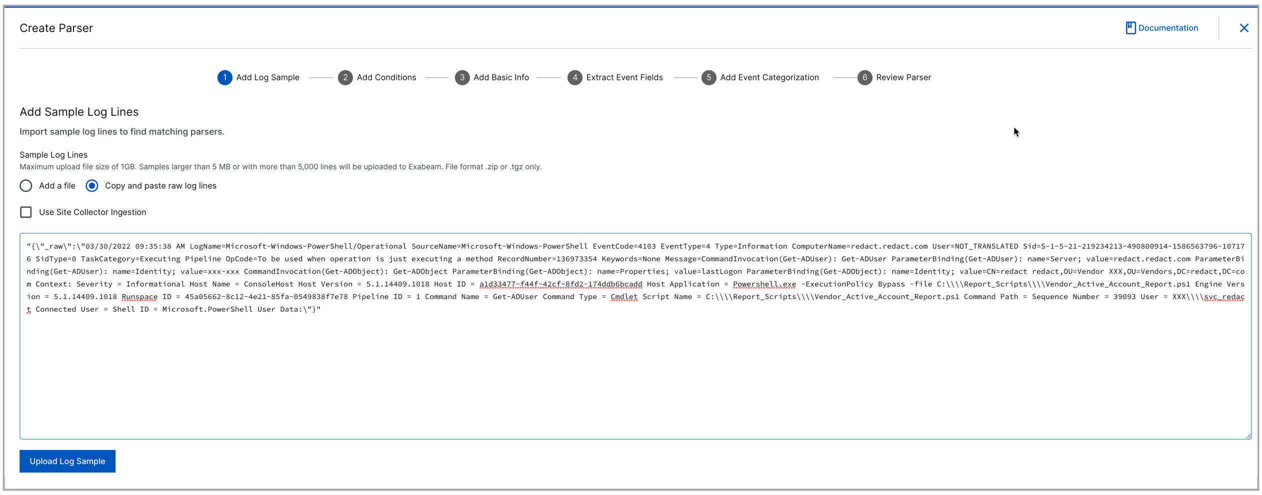
+ New Parser

□	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

Documentation

1 Add Log Sample — 2 Add Conditions — 3 Add Basic Info — 4 Extract Event Fields — 5 Add Event Categorization — 6 Review 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

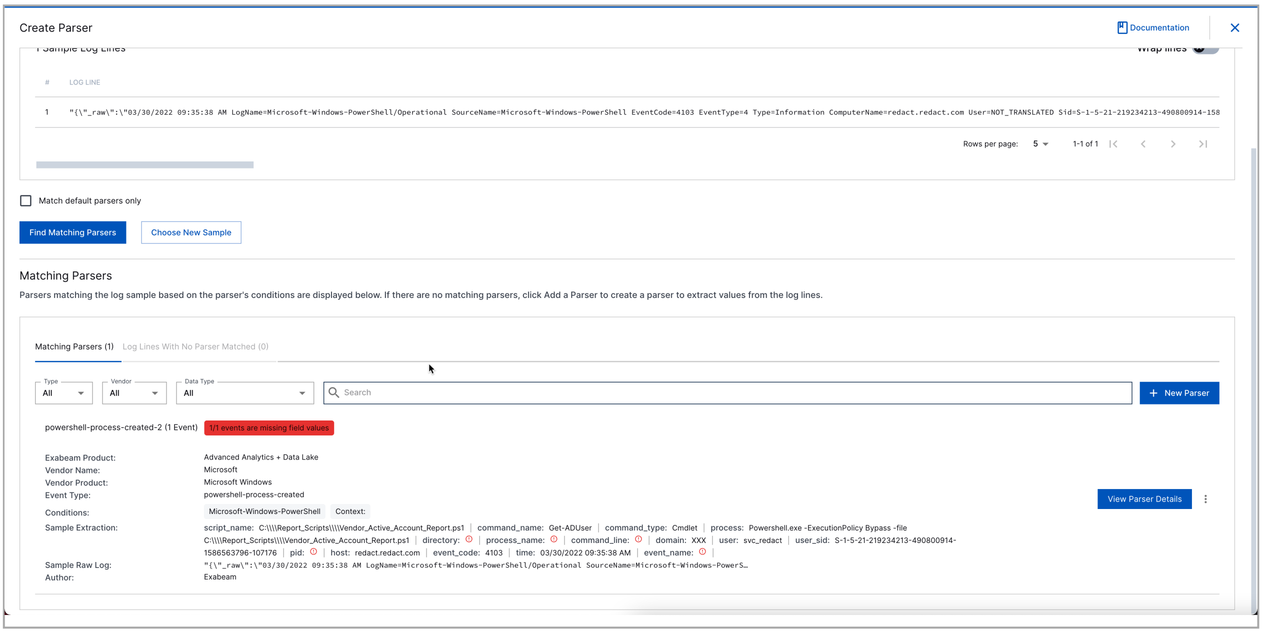
```
"{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-10717
6 SidType=0 TaskCategory=Executing 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 ParameterBi
nding(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=co
m 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 Vers
ion = 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_redac
t Connected User = Shell ID = Microsoft.PowerShell User Data:\"}"
```

Upload Log Sample

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

Documentation

Sample Log Lines

Wrap lines

#	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

Find Matching Parsers | Choose New Sample

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.

Matching Parsers (1) | Log Lines With No Parser Matched (0)

Type: All | Vendor: All | Data Type: All | Search | + New Parser

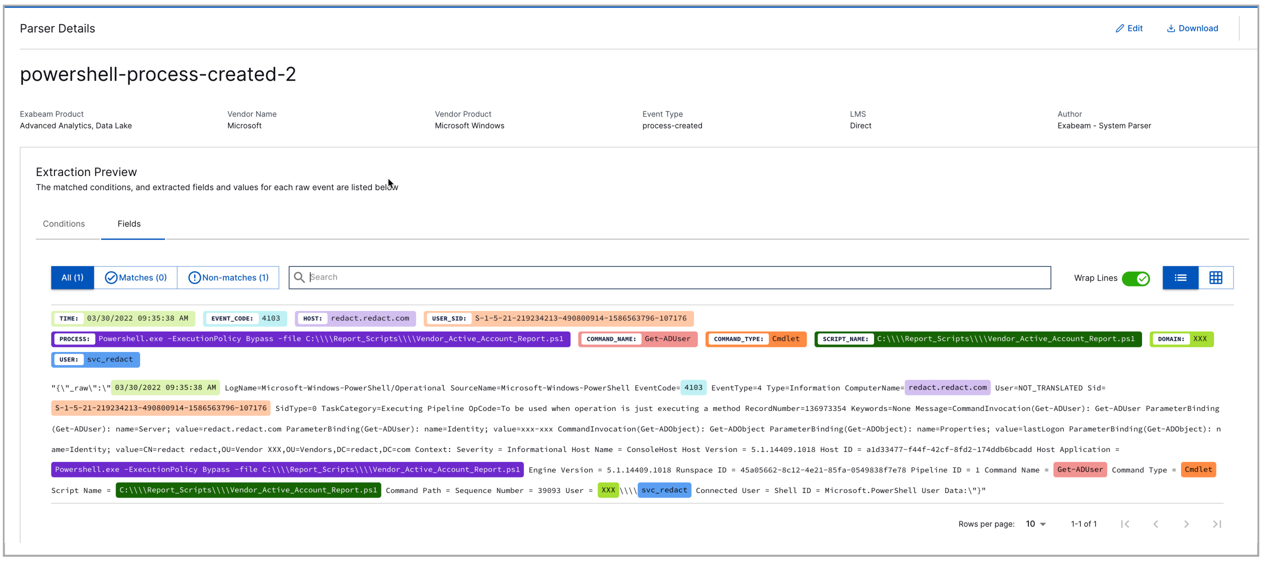
powershell-process-created-2 (1 Event) 1/1 events are missing field values

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

View Parser Details

New Parser option

Validate that all the fields you need are populated: `src_ip`, `dest_host`, `user`, etc.



Parser Details

Edit | Download

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

Conditions | Fields

All (1) | Matches (0) | Non-matches (1) | Search | Wrap Lines

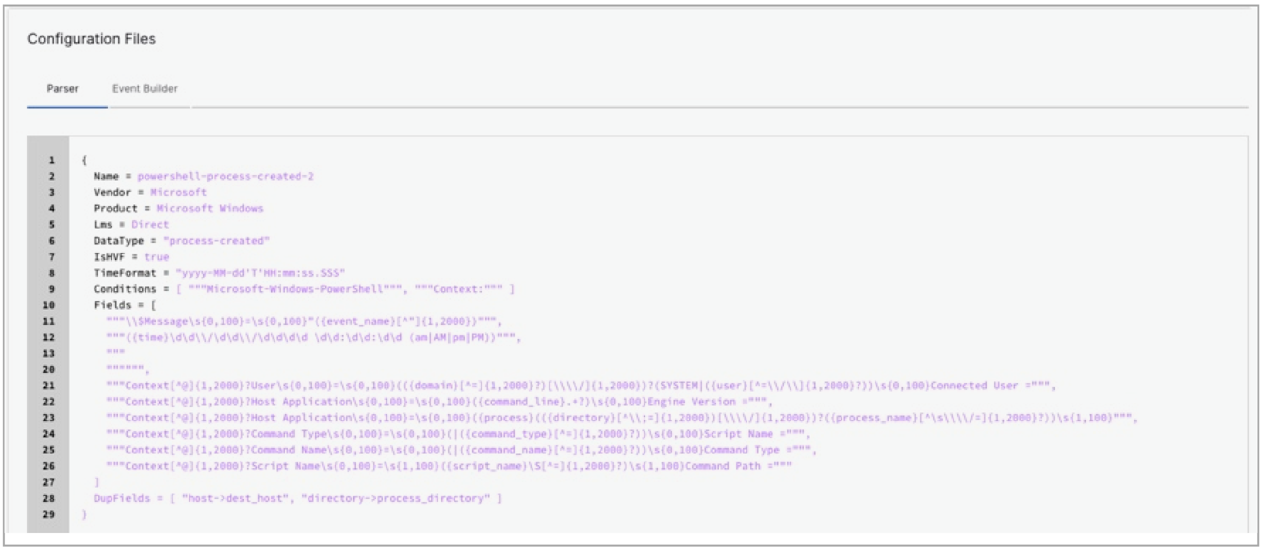
```
TIME: 03/30/2022 09:35:38 AM   EVENT_CODE: 4103   HOST: redact.redact.com   USER_SID: S-1-5-21-219234213-490800914-1586563796-107176
PROCESS: Powershell.exe -ExecutionPolicy Bypass -file C:\\\\Report_Scripts\\\\Vendor_Active_Account_Report.ps1   COMMAND_NAME: Get-ADUser   COMMAND_TYPE: Cmdlet   SCRIPT_NAME: 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

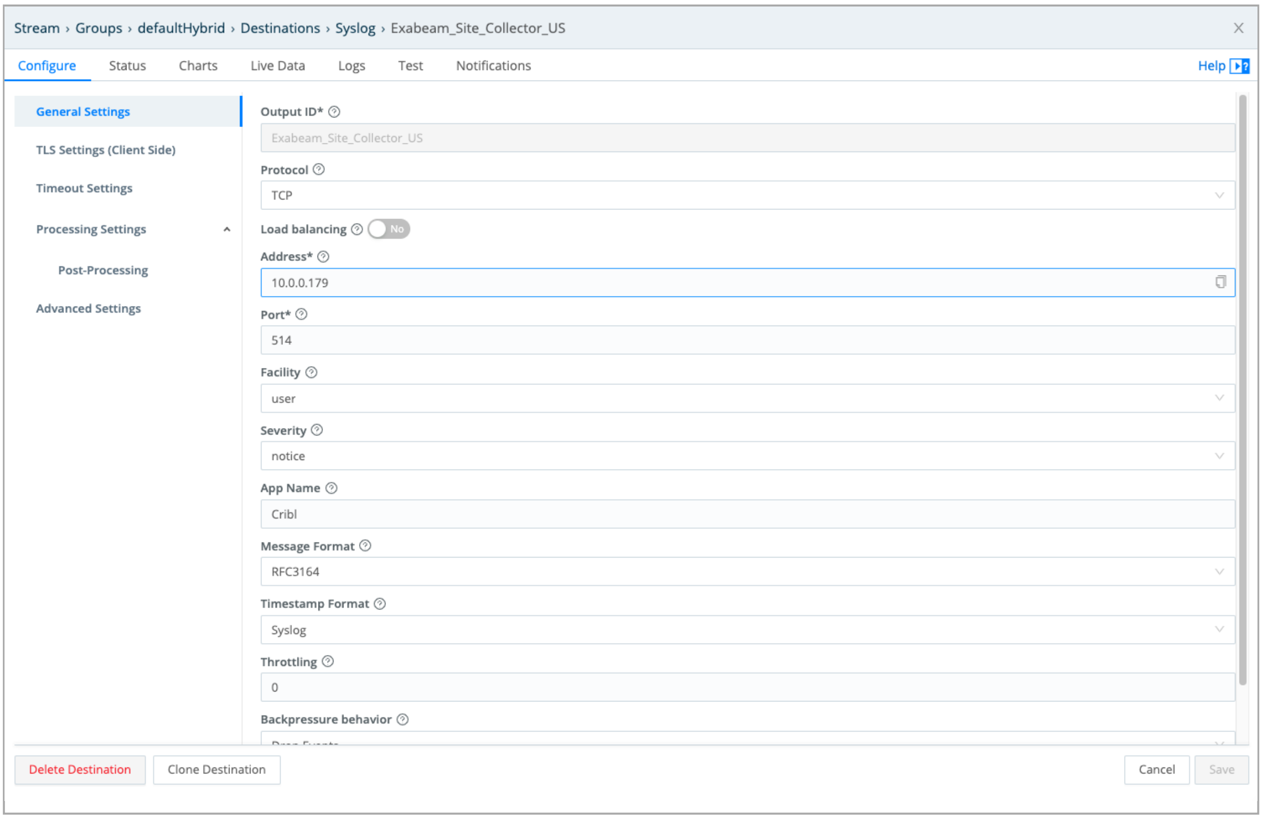
Our final test is to send a single event to Exabeam, and validate the results in Exabeam's Data Lake or Advanced Analytics.

> 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. In Cribl Stream, select **Data** > **Destinations** > **Syslog**.

2. Click **+ Add New** to configure and save a new Syslog Destination to export data to Exabeam.

   Configure the **Address**, **Port**, **Message foramt**, **Timestamp foramt**, and other options to match your Exabeam implementation.



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

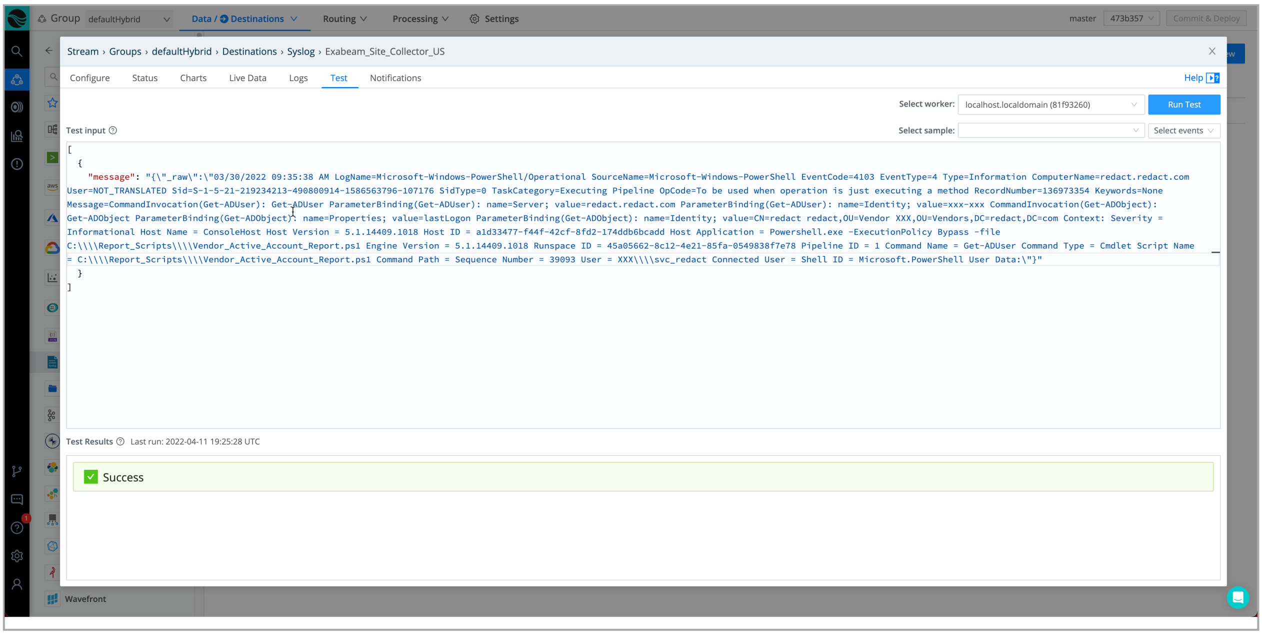
Timestamp Format: Syslog

Throttling: 0

Backpressure behavior: Drop Events

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.



Select worker: localhost.localdomain (81f93260)

Select sample:

Test input

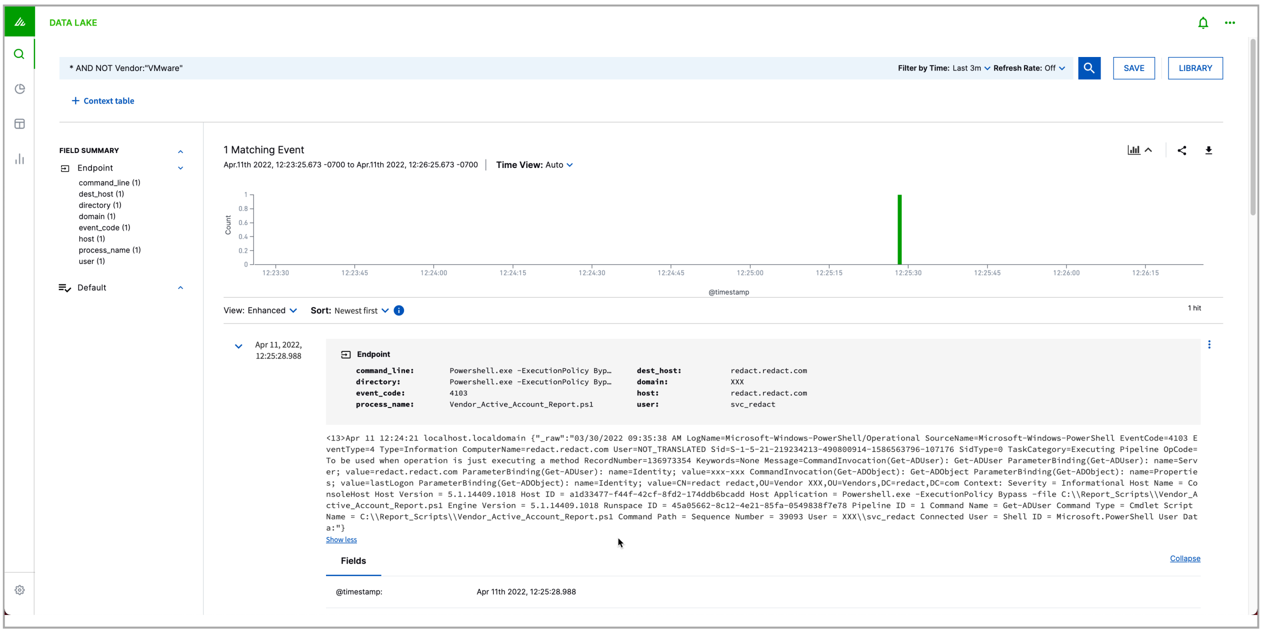
```
[
  {
    "message": "{\"_raw\":\"03/30/2022 09:35:38 AM LogName=Microsoft-Windows-PowerShell/Operational SourceName=Microsoft-Windows-PowerShell EventCode=4103 EventType=4 Type=Information ComputerName=redact.redact.com
User=NOT_TRANSLATED Sid=S-1-5-21-219234213-490800914-1586563796-107176 SidType=0 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

Endpoint
command_line (1)
dest_host (1)
directory (1)
domain (1)
event_code (1)
host (1)
process_name (1)
user (1)

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

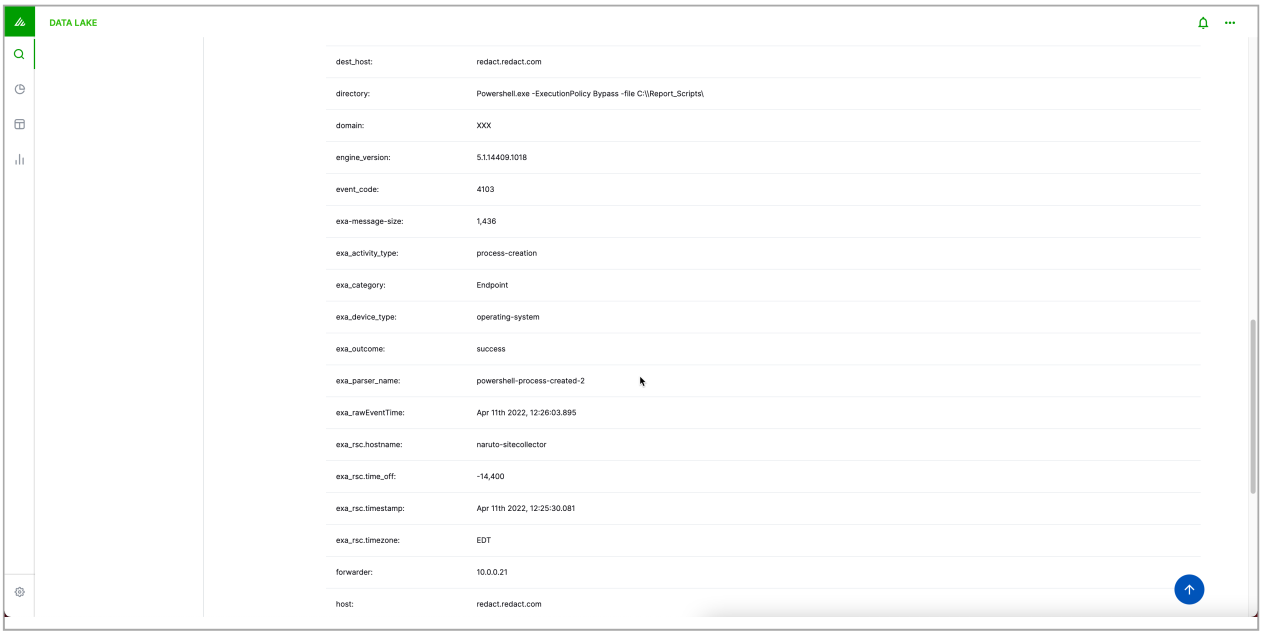
```
command_line:    Powershell.exe -ExecutionPolicy Byp…    dest_host:    redact.redact.com
directory:       Powershell.exe -ExecutionPolicy Byp…    domain:       XXX
event_code:      4103                                    host:         redact.redact.com
process_name:    Vendor_Active_Account_Report.ps1        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

;

11.2.8. 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/splunk_app_stream/
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:

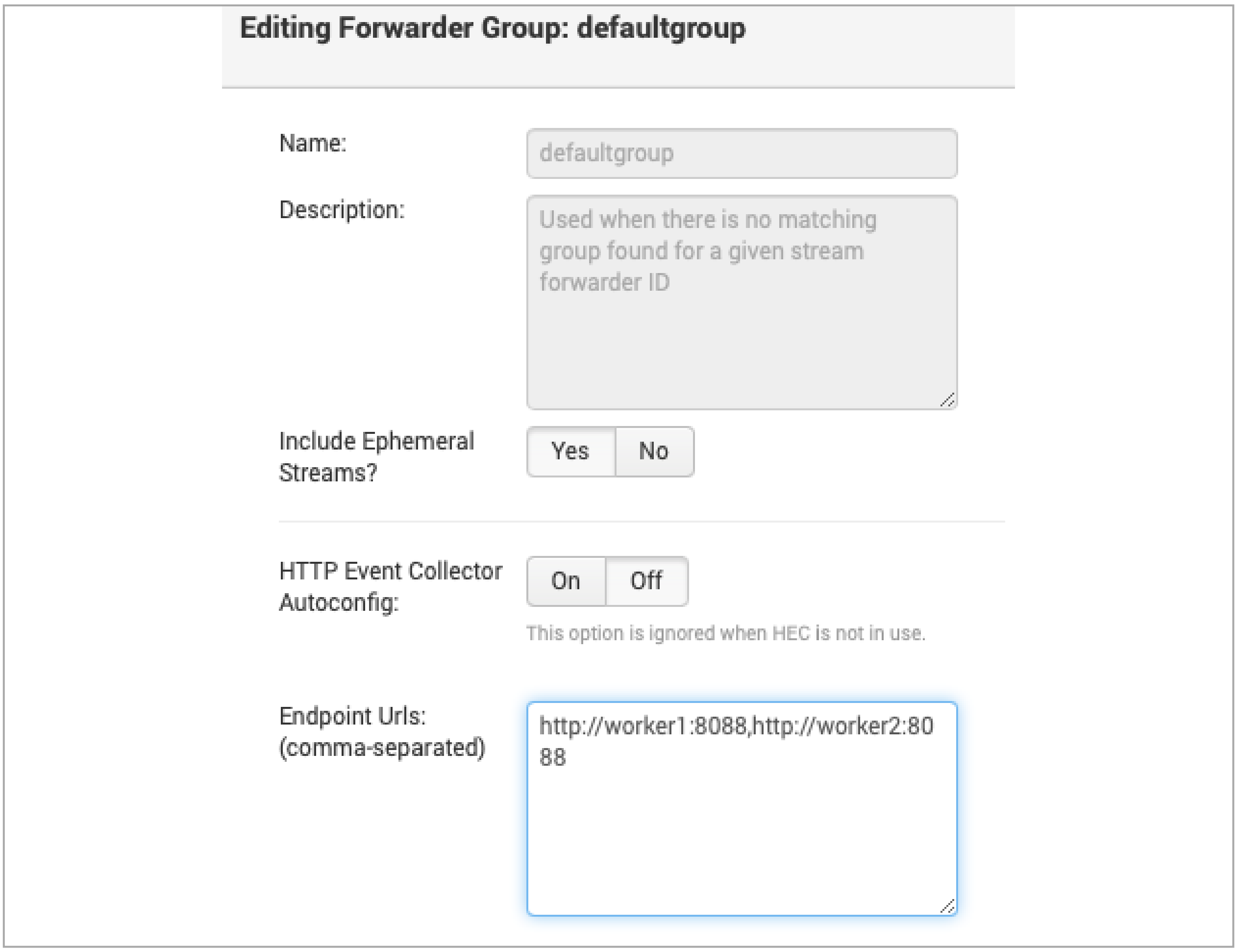
- In the Stream App, navigate to **Configuration** > **Distributed Forwarder Management**.
- Click the **Install Stream Forwarders** button.
- In the resulting modal, under the text `To get data from other machines, run this command on your data source machine`, copy the `curl` command. E.g.:

  ```
  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:

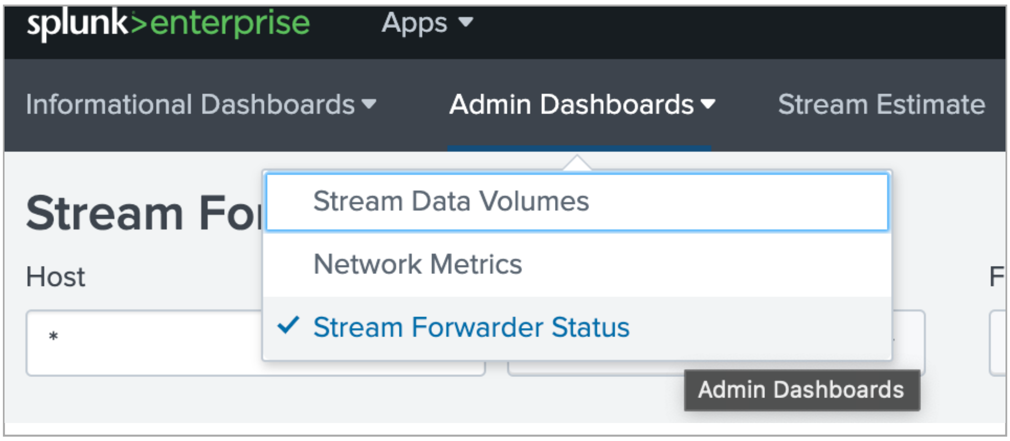
```
[streamfwd]
httpEventCollectorToken = <HEC_token_from_your_Cribl_Stream_Source>
```

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.

;

11.2.9. 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.

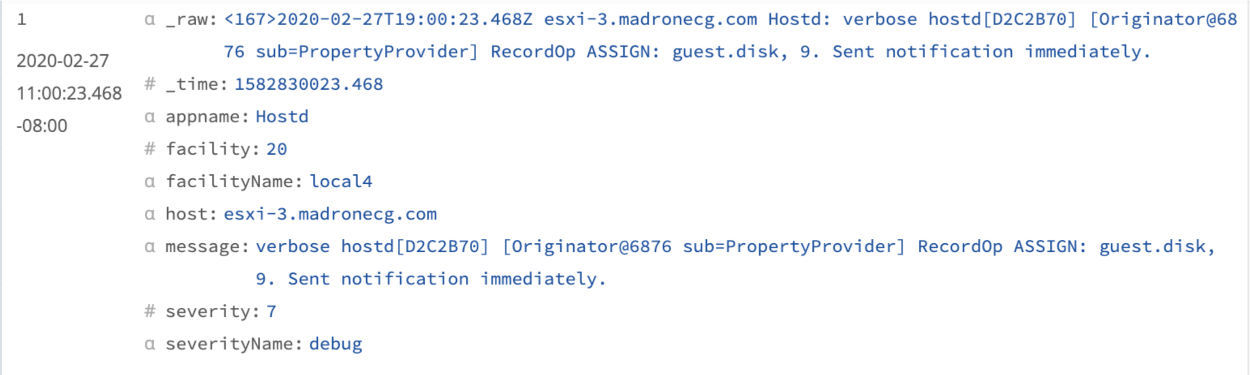
Processing syslog in Cribl Stream allows you to readily address these common challenges of ingesting data from syslog senders:

- **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).

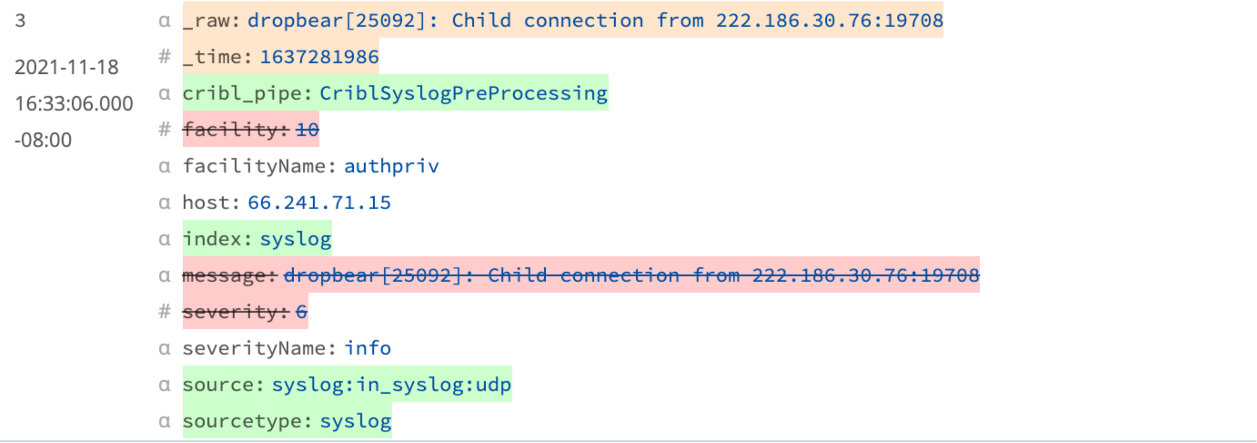


```
1              α _raw: <167>2020-02-27T19:00:23.468Z esxi-3.madronecg.com Hostd: verbose hostd[D2C2B70] [Originator@68
                       76 sub=PropertyProvider] RecordOp ASSIGN: guest.disk, 9. Sent notification immediately.
2020-02-27     # _time: 1582830023.468
11:00:23.468   α appname: Hostd
-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:

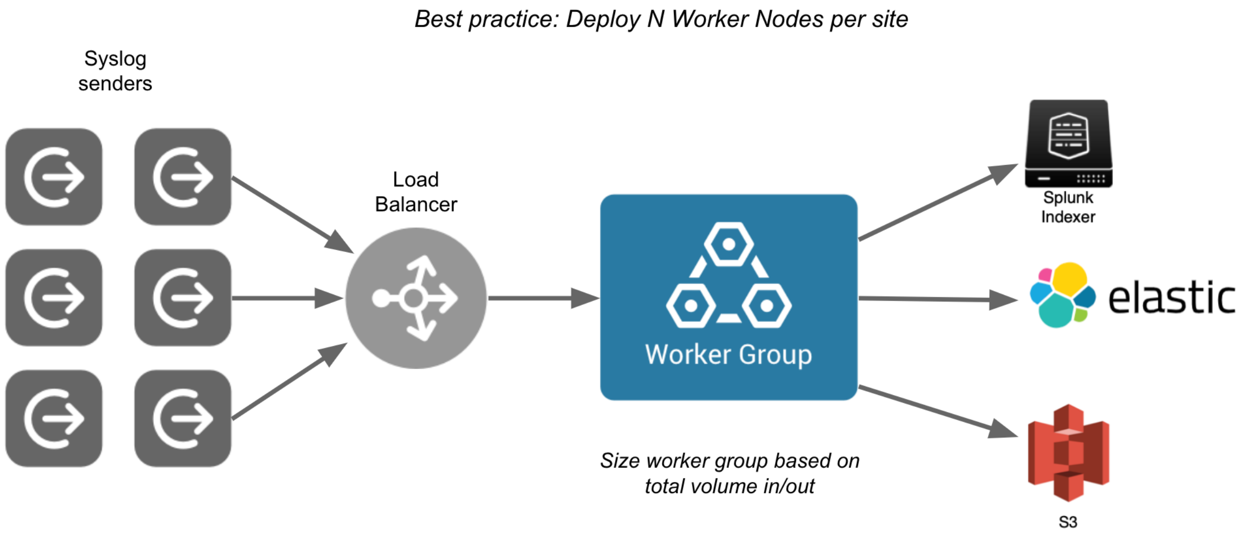
- Extracts the essentials.
- Removes the redundancies.
- Adds a new field to identify the Cribl Stream Pipeline (which we're about to build).
- Adds metadata that the Destination needs.
- Shrinks the outbound `_raw` payload to just its `message` component.

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
Splunk Indexer
elastic
S3
Size worker group based on total volume in/out

Example architecture for a single site or location

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.

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.)

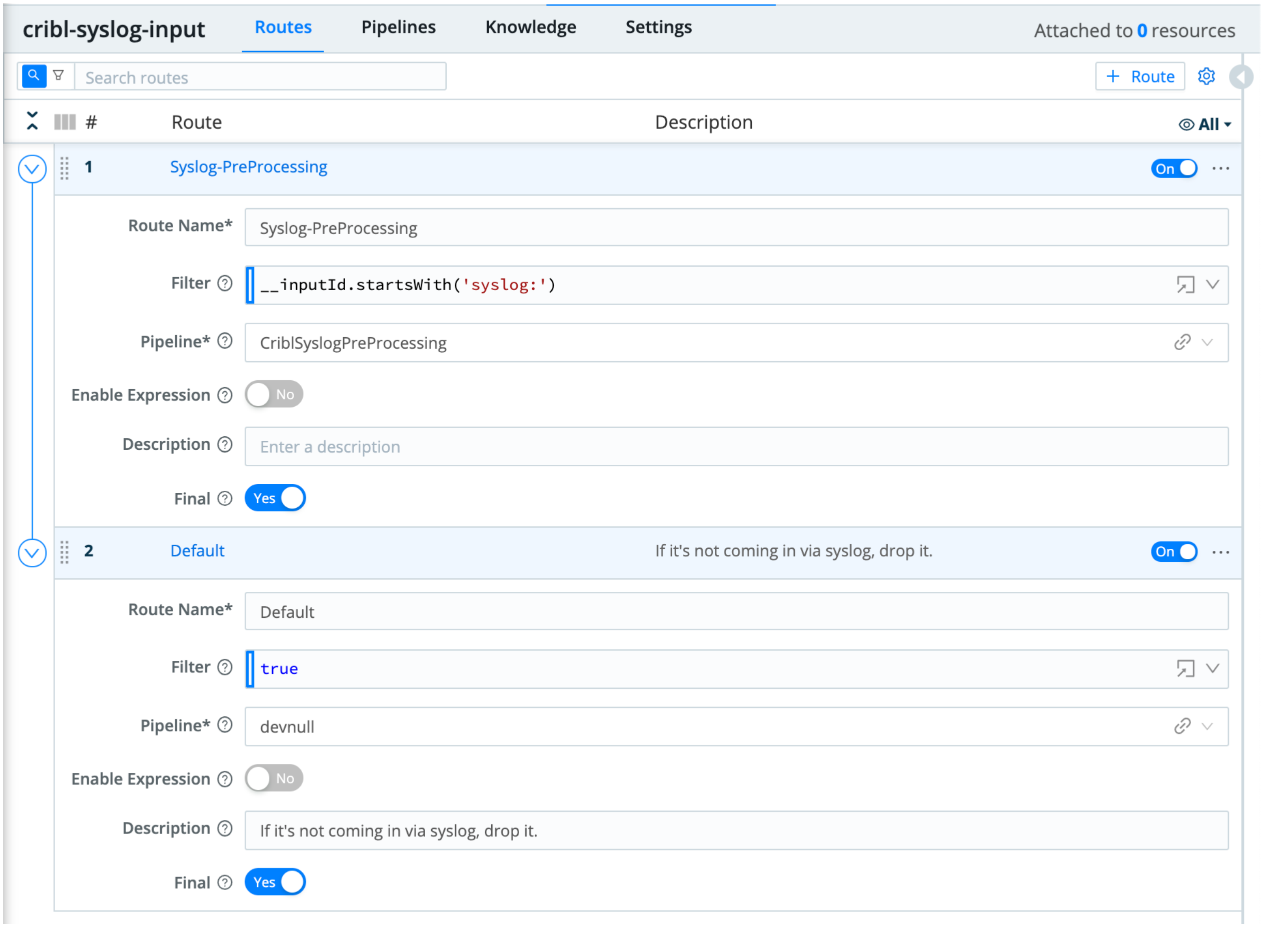
Importing the Pre-Processing Pipeline

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.



1 Syslog-PreProcessing On

Route Name* Syslog-PreProcessing

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

Pipeline* CriblSyslogPreProcessing

Enable Expression No

Description Enter a description

Final Yes

2 Default If it's not coming in via syslog, drop it. On

Route Name* Default

Filter `true`

Pipeline* devnull

Enable Expression No

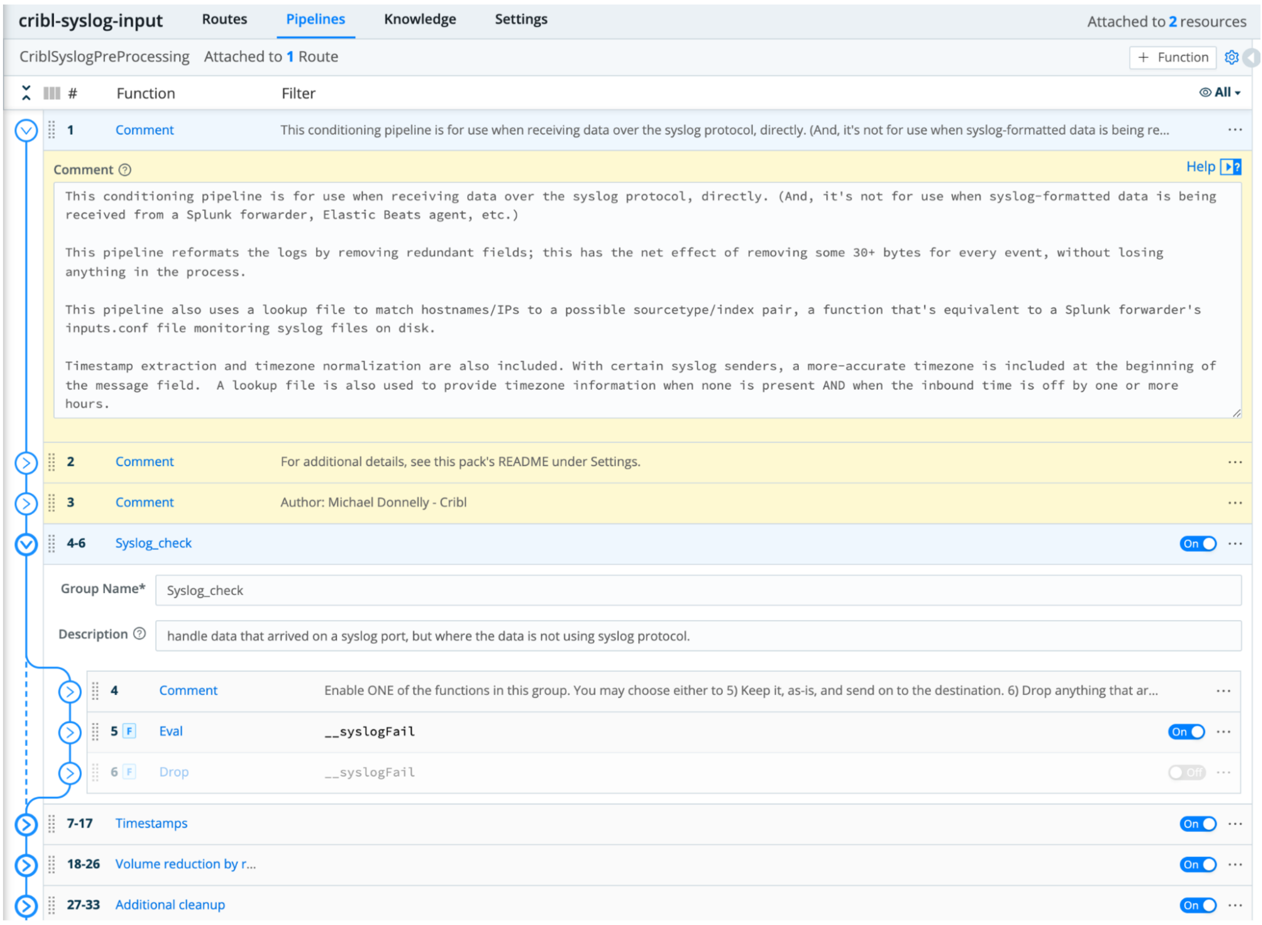
Description If it's not coming in via syslog, drop it.

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.


cribl-syslog-input | Routes | Pipelines | Knowledge | Settings | Attached to 2 resources

CriblSyslogPreProcessing Attached to 1 Route

#	Function	Filter
1	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 re...

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.
```

#	Function	Filter	
2	Comment	For additional details, see this pack's README under Settings.	
3	Comment	Author: Michael Donnelly - Cribl	
4-6	Syslog_check		On

Group Name*: Syslog_check

Description: handle data that arrived on a syslog port, but where the data is not using syslog protocol.

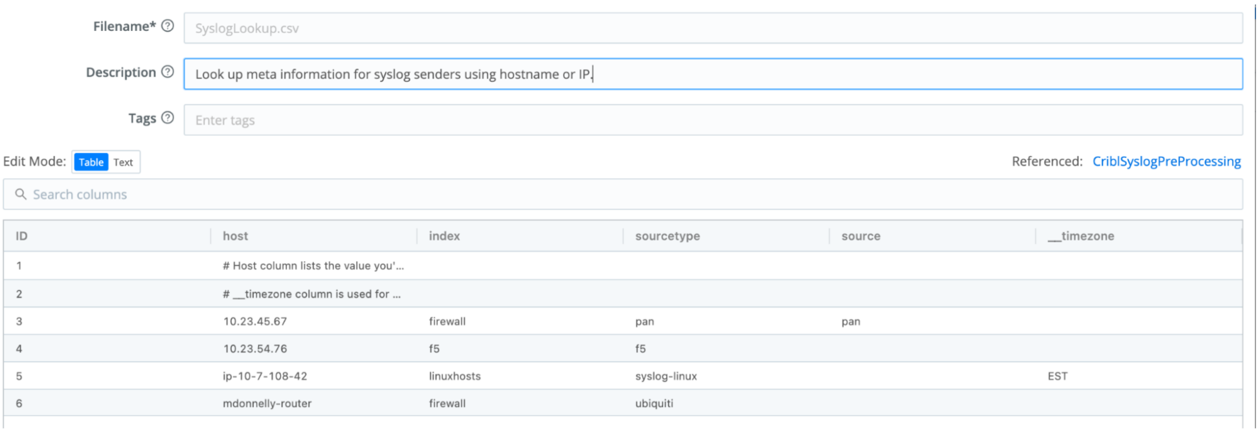
#	Function	Filter	
4	Comment	Enable ONE of the functions in this group. You may choose either to 5) Keep it, as-is, and send on to the destination. 6) Drop anything that ar...	
5	Eval	`__syslogFail`	On
6	Drop	`__syslogFail`	Off
7-17	Timestamps		On
18-26	Volume reduction by r...		On
27-33	Additional cleanup		On

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**.


Filename*: SyslogLookup.csv

Description: Look up meta information for syslog senders using hostname or IP.

Tags: Enter tags

Edit Mode: Table Text

Referenced: CriblSyslogPreProcessing

ID	host	index	sourcetype	source	__timezone
1	# Host column lists the value you'...				
2	# __timezone column is used for ...				
3	10.23.45.67	firewall	pan	pan	
4	10.23.54.76	f5	f5		
5	ip-10-7-108-42	linuxhosts	syslog-linux		EST
6	mdonnelly-router	firewall	ubiquiti		

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 **+ New 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 **+ New 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 **+ Add New** to open a **New Source** modal.

Configure the fields and options as follows:

General Settings

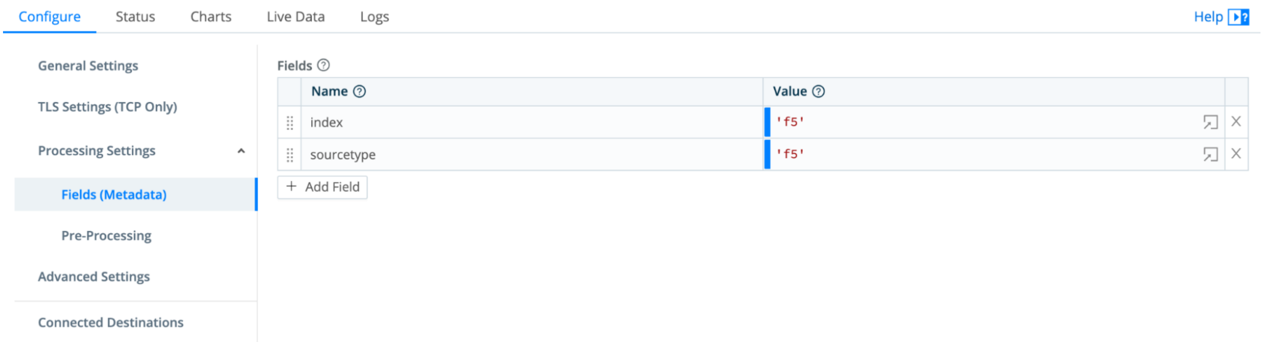
- **Input ID**: This field specifies the name for the Source, which will appear in the `__inputId` field as well as on Monitoring pages that show Source information. Common examples include `in_syslog_cisco_switch` for Cisco switches, `in_syslog_f5` for F5 load balancers, and so on.
- **UDP Port** and **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.

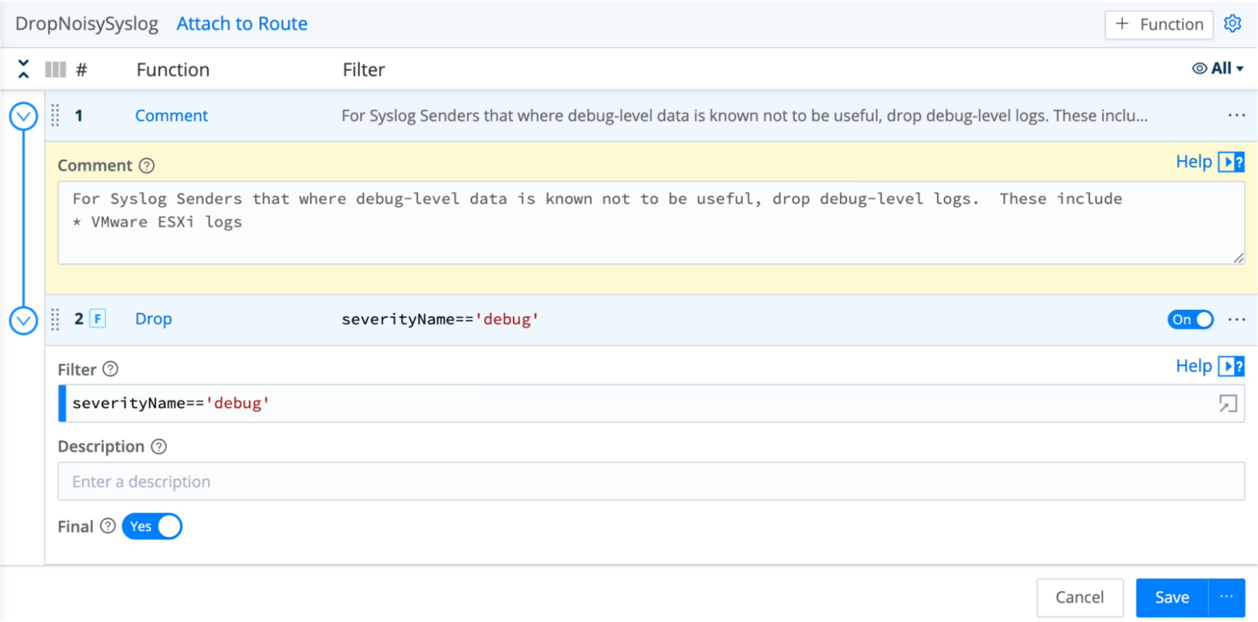
Use Case A: QuickConnect a Syslog Sender to a Source

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 **+ 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.



For Syslog Senders that where debug-level data is known not to be useful, drop debug-level logs. These inclu...

```
For Syslog Senders that where debug-level data is known not to be useful, drop debug-level logs.  These include
* VMware ESXi logs
```

```
severityName=='debug'
```

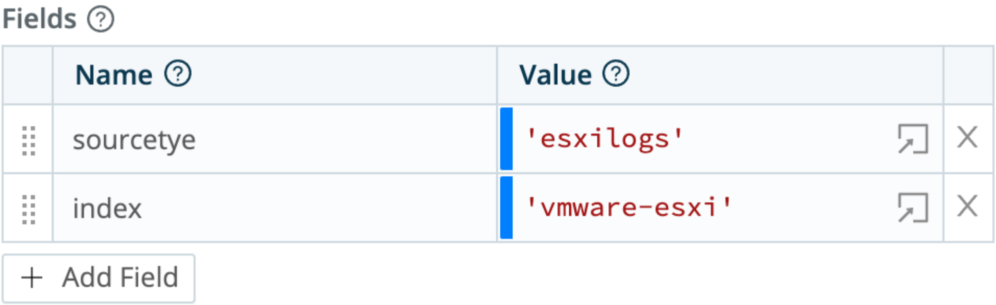
Pipeline to remove debug events

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 **+ New 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`.



Fields

Name	Value
sourcetye	'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 **+ 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.

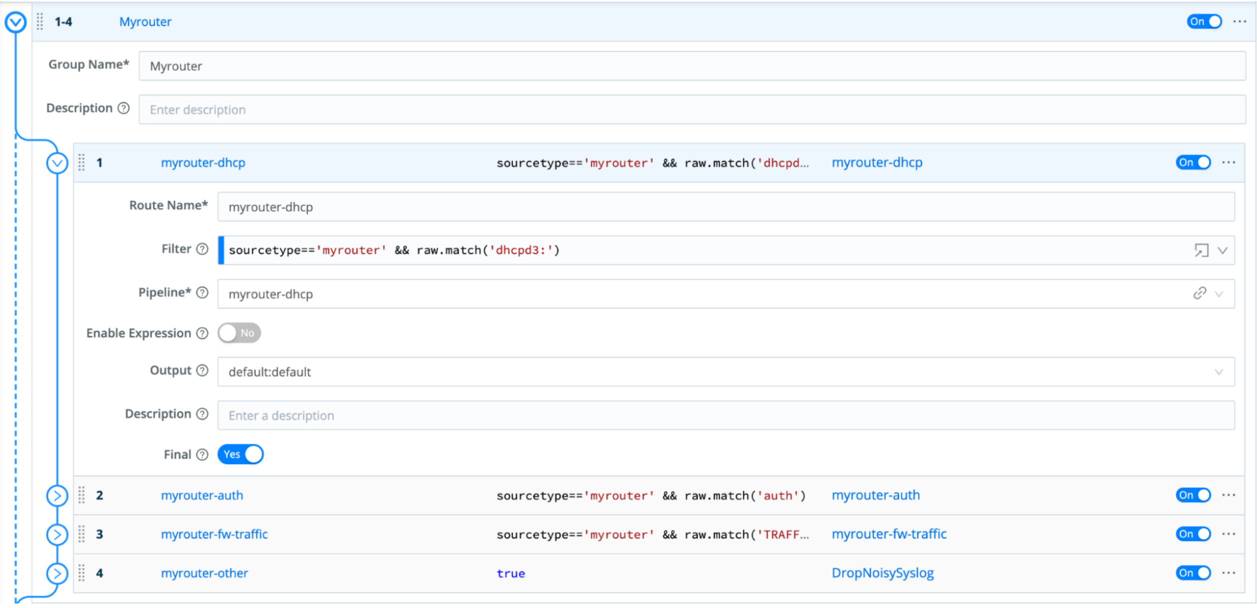
5. Click the `myrouter-dhcp` Route's ••• (Options) menu, then select **Group Actions** > **Create Group**.

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:



1-4 Myrouter On

Group Name* Myrouter

Description Enter description

1 myrouter-dhcp `sourcetype=='myrouter' && raw.match('dhcpd…` myrouter-dhcp On

Route Name* myrouter-dhcp

Filter `sourcetype=='myrouter' && raw.match('dhcpd3:')`

Pipeline* myrouter-dhcp

Enable Expression No

Output default:default

Description Enter a description

Final Yes

2 myrouter-auth `sourcetype=='myrouter' && raw.match('auth')` myrouter-auth On

3 myrouter-fw-traffic `sourcetype=='myrouter' && raw.match('TRAFF…` myrouter-fw-traffic On

4 myrouter-other `true` DropNoisySyslog On

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.

;

11.2.10. Sumo Logic/Webhook Integration

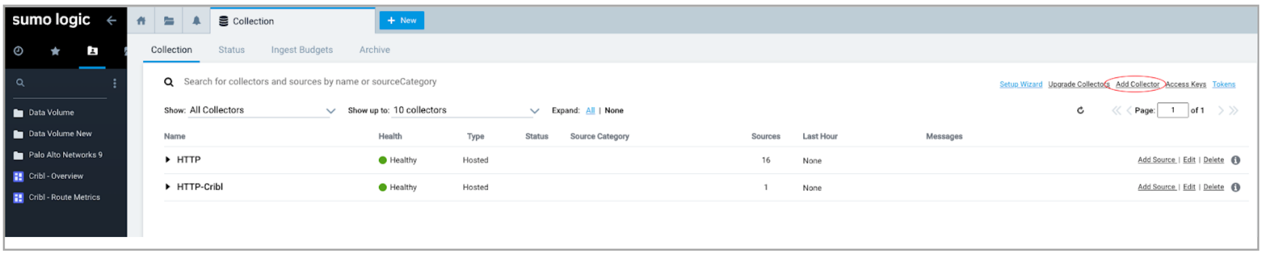
You can configure Cribl Stream to send raw event data through a Webhook Destination to Sumo Logic's log management and analytics service.

Before you begin, you need the `Manage Collectors` role capability in Sumo Logic.

Prepare Sumo Logic

1. In Sumo Logic, navigate to **Manage Data** > **Collection**.

2. If you already have a Hosted Collector, select it and skip to Step 4. If you do not already have a Hosted Collector, click **Add Collector**.



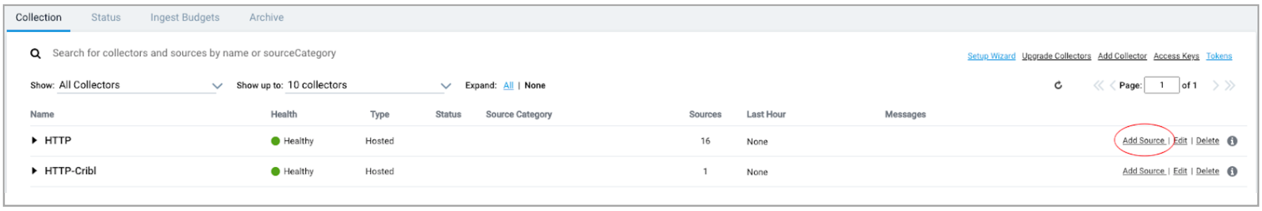
Show: All Collectors　Show up to: 10 collectors　Expand: All | None　Page: 1 of 1

Name	Health	Type	Status	Source Category	Sources	Last Hour	Messages	
▸ HTTP	● Healthy	Hosted			16	None		Add Source | Edit | Delete
▸ HTTP-Cribl	● Healthy	Hosted			1	None		Add Source | Edit | Delete

Adding a Hosted Collector

3. Select **Hosted Collector** and fill in the form.

4. Click **Add Source**.



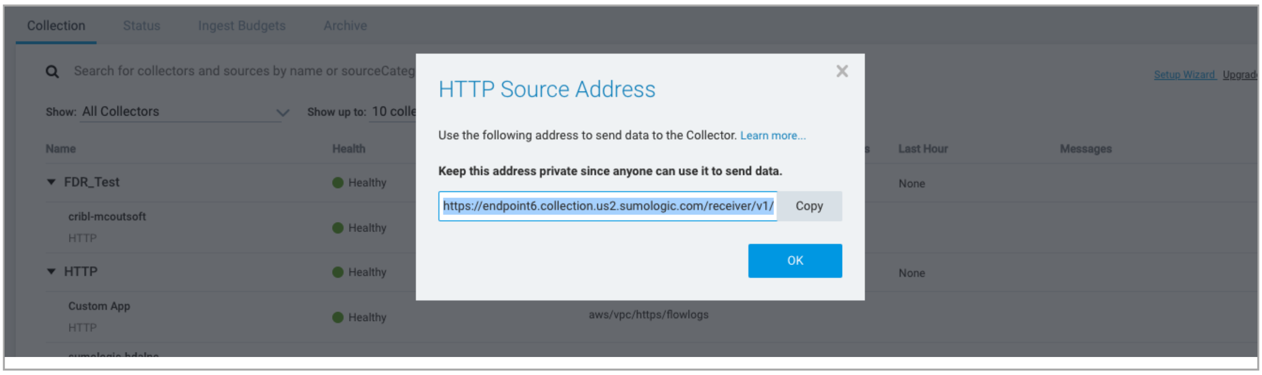
Show: All Collectors　Show up to: 10 collectors　Expand: All | None　Page: 1 of 1

Name	Health	Type	Status	Source Category	Sources	Last Hour	Messages	
▸ HTTP	● Healthy	Hosted			16	None		Add Source | Edit | Delete
▸ HTTP-Cribl	● Healthy	Hosted			1	None		Add Source | Edit | Delete

Adding a Source for your Hosted Collector

5. Select **HTTP Logs & Metrics**. Fill in the form and click **Save**.

6. Sumo Logic then displays the HTTP Source URL, which you will need when configuring the Webhook Destination in Cribl Stream.



HTTP Source Address

Use the following address to send data to the Collector. Learn more...

Keep this address private since anyone can use it to send data.

https://endpoint6.collection.us2.sumologic.com/receiver/v1/

The HTTP Source URL

7. Click **Copy** to capture the URL.

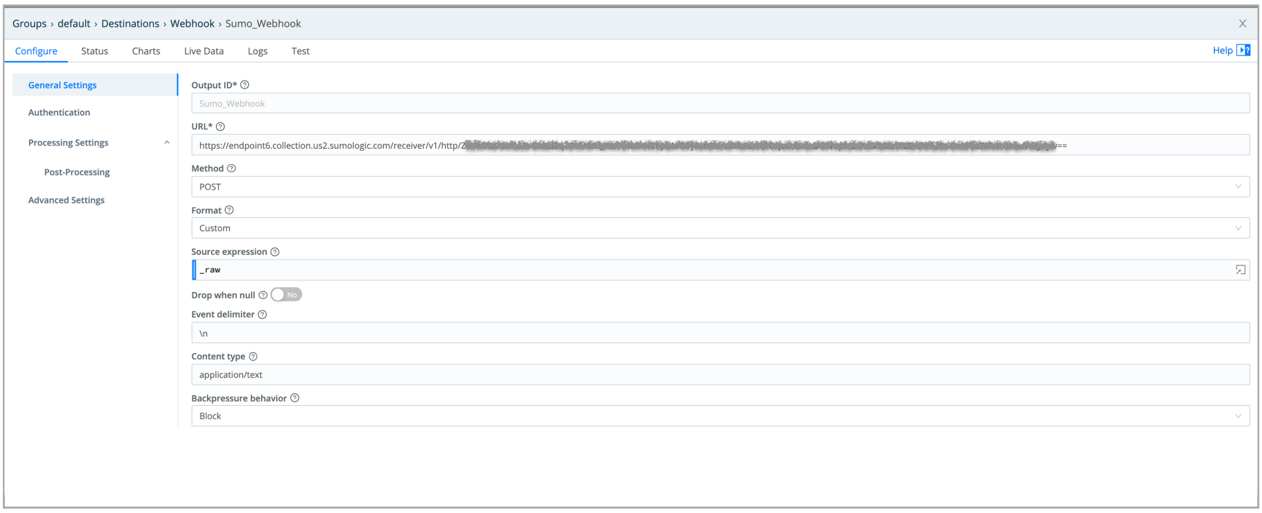
8. Click **Next** to finish configuring the HTTP Source.

Configure Cribl Stream's Webhook Destination

From the top nav of a Cribl Stream instance or Group, 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.

In the **Configure** > **General Settings** tab, enter or select the following values:

- **URL**: Enter the Sumo Logic HTTP Source URL.
- **Method**: `POST`
- **Format**: `Custom`
- **Source Expression** = `_raw`. As needed, add more fields (e.g., `host`) for Webhook to include in the raw output.
- **Content type**: `application/text`



Output ID*: Sumo_Webhook

URL*: https://endpoint6.collection.us2.sumologic.com/receiver/v1/http/Z[redacted]==

Method: POST

Format: Custom

Source expression: `_raw`

Drop when null: No

Event delimiter: \n

Content type: application/text

Backpressure behavior: Block

Webhook Destination General Settings

Next, in the **Authentication** tab, make sure that **Authentication type** is set to `None`.

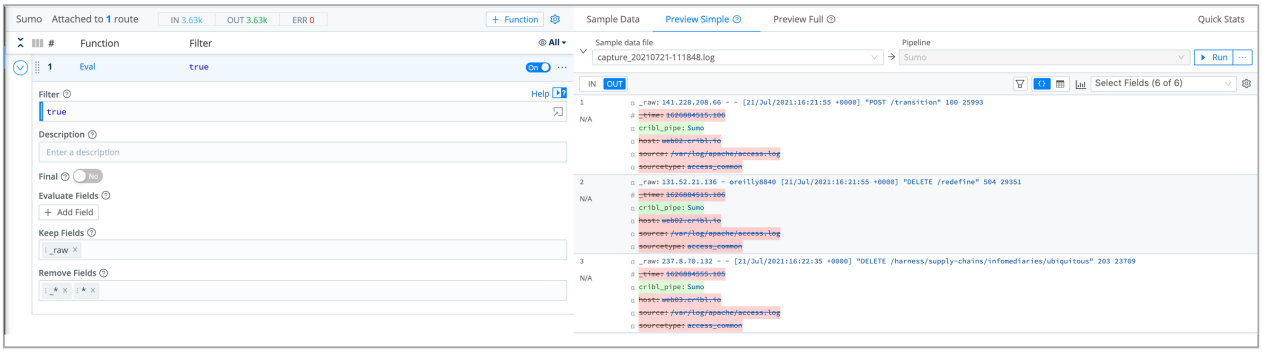
In the **Post-Processing** tab, remove `cribl_pipe` from **System Fields**.

Click **Save**, then **Commit & Deploy**.

Create a Pipeline

Create a Cribl Stream Pipeline to remove all except the **Source Expression** fields. Add an Eval Function with the following values:

- **Keep Fields**: Specify the same field(s) you specified in **Source Expression** when you created the Webhook Destination (e.g., `_raw`, `host`).
- **Remove Fields**: `_*` and `*`, to remove all other fields.



1 Eval true

Filter: `true`

Description: Enter a description

Final: No

Evaluate Fields

Keep Fields: `_raw`

Remove Fields: `_*` `*`

Sample data file: capture_20210721-111848.log → Pipeline: Sumo

1 N/A
_raw: 141.228.208.66 - - [21/Jul/2021:16:21:55 +0000] "POST /transition" 100 25993
~~_time: 1626884515.106~~
cribl_pipe: Sumo
~~host: web02.cribl.io~~
~~source: /var/log/apache/access.log~~
~~sourcetype: access_common~~

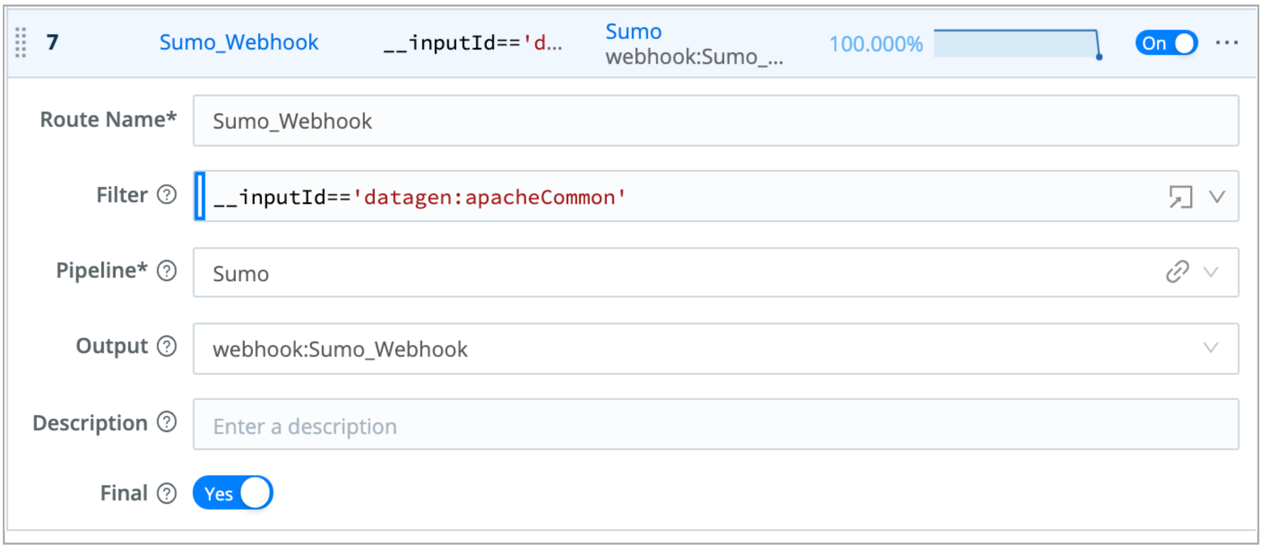
2 N/A
_raw: 131.52.21.136 - oreilly8840 [21/Jul/2021:16:21:55 +0000] "DELETE /redefine" 504 29351
~~_time: 1626884515.106~~
cribl_pipe: Sumo
~~host: web02.cribl.io~~
~~source: /var/log/apache/access.log~~
~~sourcetype: access_common~~

3 N/A
_raw: 237.8.70.132 - - [21/Jul/2021:16:22:35 +0000] "DELETE /harness/supply-chains/infomediaries/ubiquitous" 203 23709
~~_time: 1626884555.105~~
cribl_pipe: Sumo
~~host: web03.cribl.io~~
~~source: /var/log/apache/access.log~~
~~sourcetype: access_common~~

Example Pipeline

Create a Route

Configure a Route similar to the example below, replacing the names/IDs shown with the actual names/IDs from your setup.



Route Name*: Sumo_Webhook

Filter: `__inputId=='datagen:apacheCommon'`

Pipeline*: Sumo

Output: webhook:Sumo_Webhook

Description: Enter a description

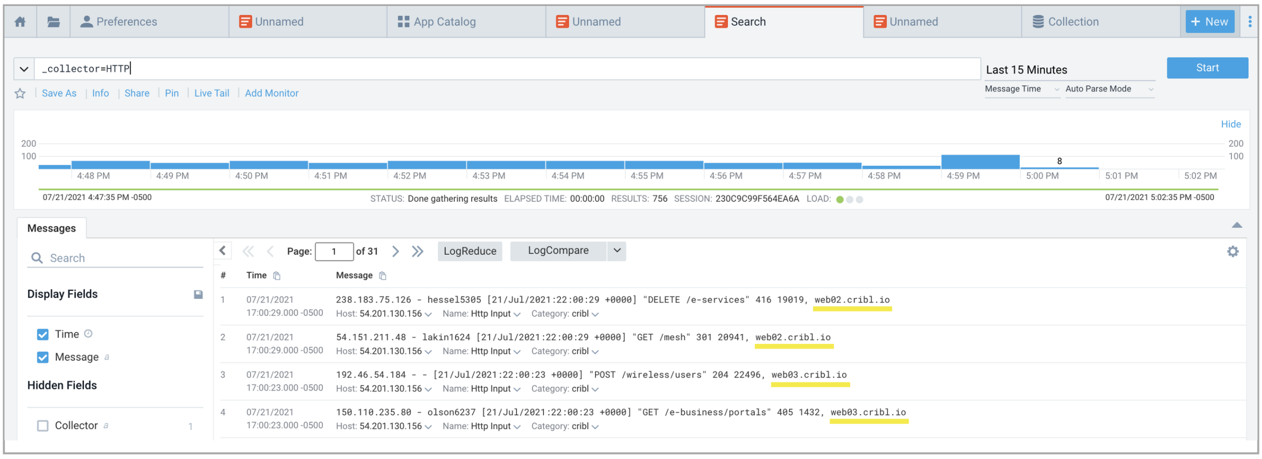
Final: Yes

Example Route

Verify that Sumo Logic Is Receiving Data

In Sumo Logic, search for `_source=<name>`, replacing `<name>` with the name of the Source you created. The search should return event data that is broken out into fields.

If your Webhook Destination specifies **Source Expression** as `_raw, host`, you should see output similar to this, where the added field (`host`) appears at the end of the `_raw` string.



Search: `_collector=HTTP`

07/21/2021 4:47:35 PM -0500 STATUS: Done gathering results ELAPSED TIME: 00:00:00 RESULTS: 756 SESSION: 230C9C99F564EA6A LOAD: 07/21/2021 5:02:35 PM -0500

```
#  Time                          Message
1  07/21/2021 17:00:29.000 -0500  238.183.75.126 - hessel5305 [21/Jul/2021:22:00:29 +0000] "DELETE /e-services" 416 19019, web02.cribl.io
                                  Host: 54.201.130.156  Name: Http Input  Category: cribl
2  07/21/2021 17:00:29.000 -0500  54.151.211.48 - lakin1624 [21/Jul/2021:22:00:29 +0000] "GET /mesh" 301 20941, web02.cribl.io
                                  Host: 54.201.130.156  Name: Http Input  Category: cribl
3  07/21/2021 17:00:23.000 -0500  192.46.54.184 - - [21/Jul/2021:22:00:23 +0000] "POST /wireless/users" 204 22496, web03.cribl.io
                                  Host: 54.201.130.156  Name: Http Input  Category: cribl
4  07/21/2021 17:00:23.000 -0500  150.110.235.80 - olson6237 [21/Jul/2021:22:00:23 +0000] "GET /e-business/portals" 405 1432, web03.cribl.io
                                  Host: 54.201.130.156  Name: Http Input  Category: cribl
```

Sumo Logic Search Results with an Added Field

;

11.2.11. 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**.

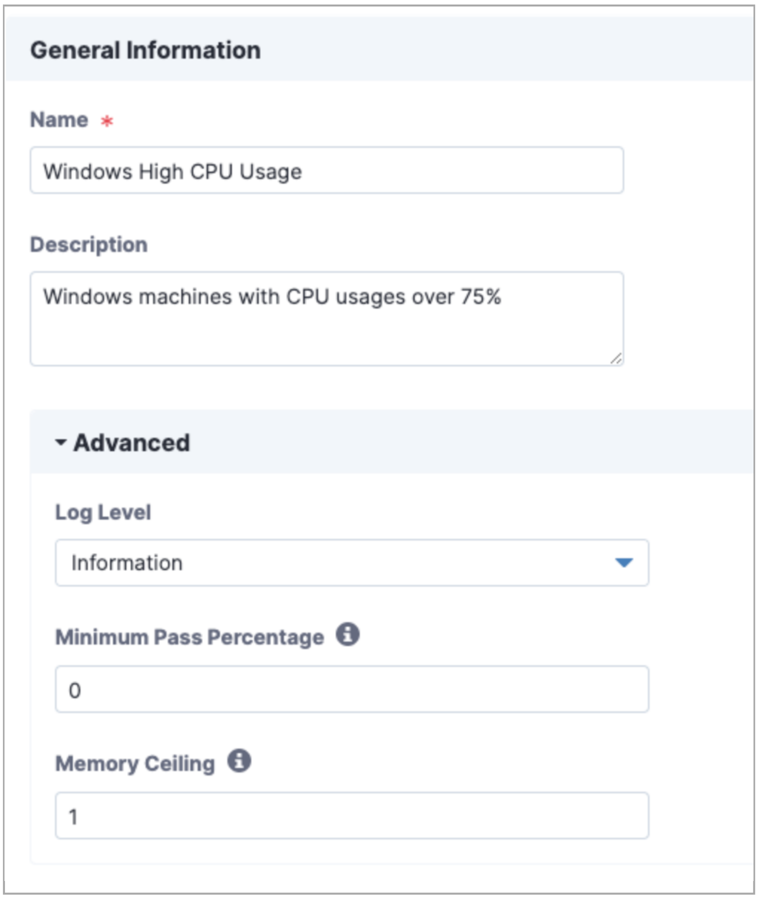
Specify General Connection Information

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

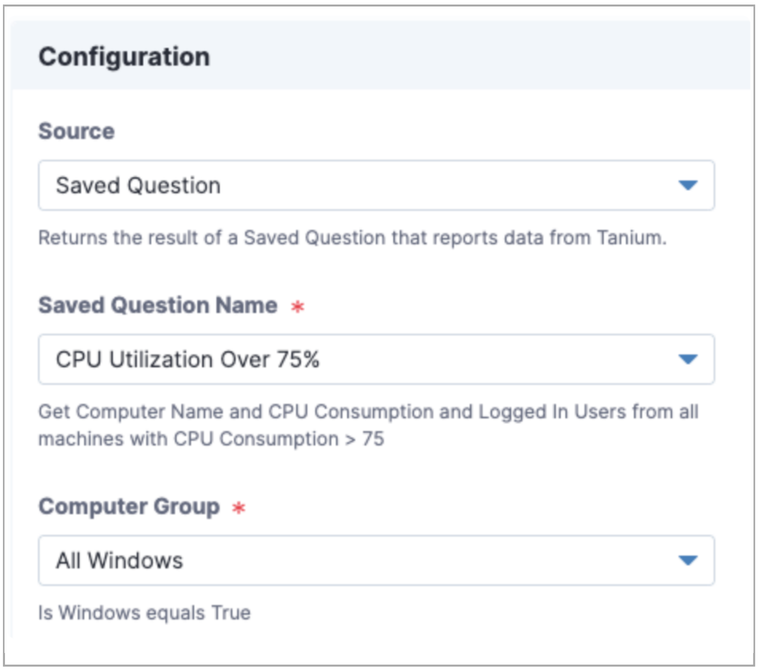
Memory Ceiling

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

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.

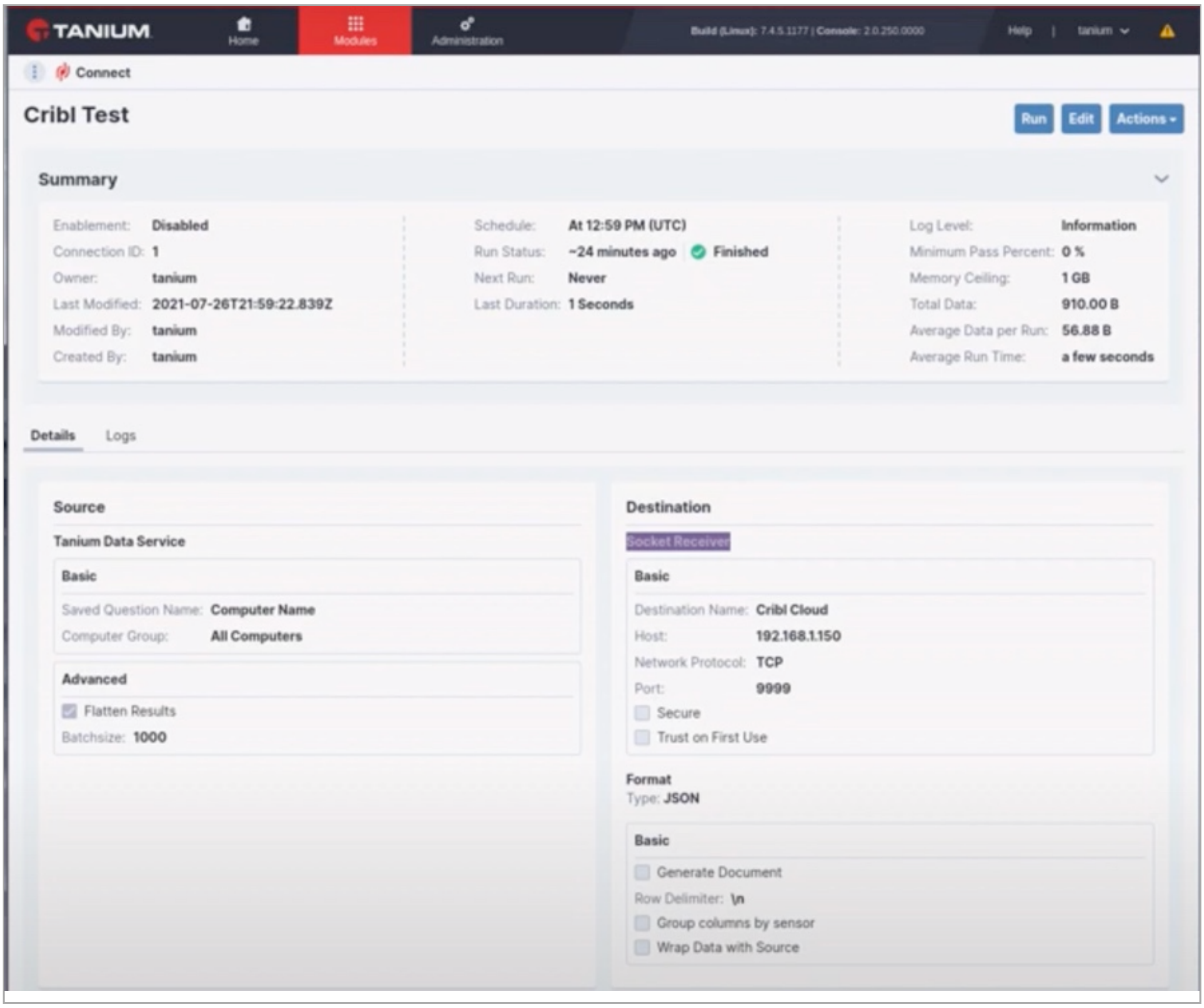
Host: Specify the destination'server host.

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

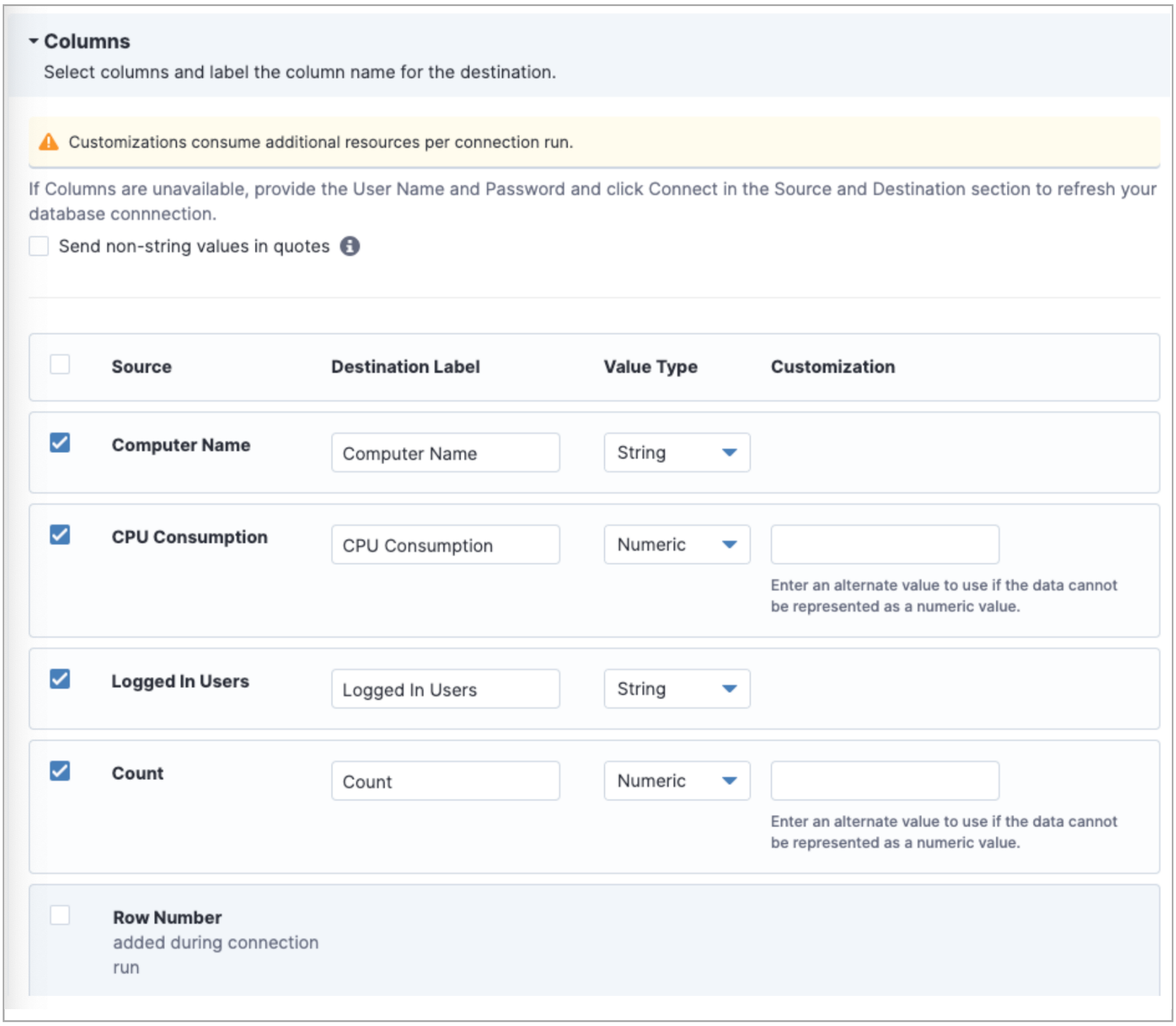
Format the Data

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

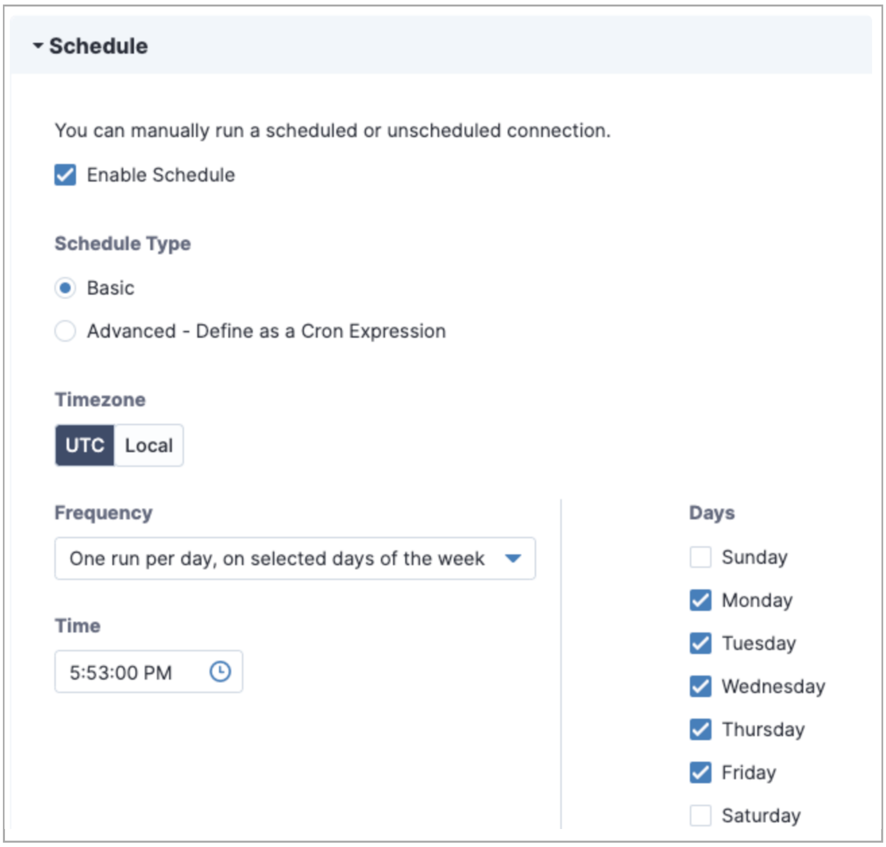
Schedule the Connection

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.

;

11.3. Lookup Applications

11.3.1. 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

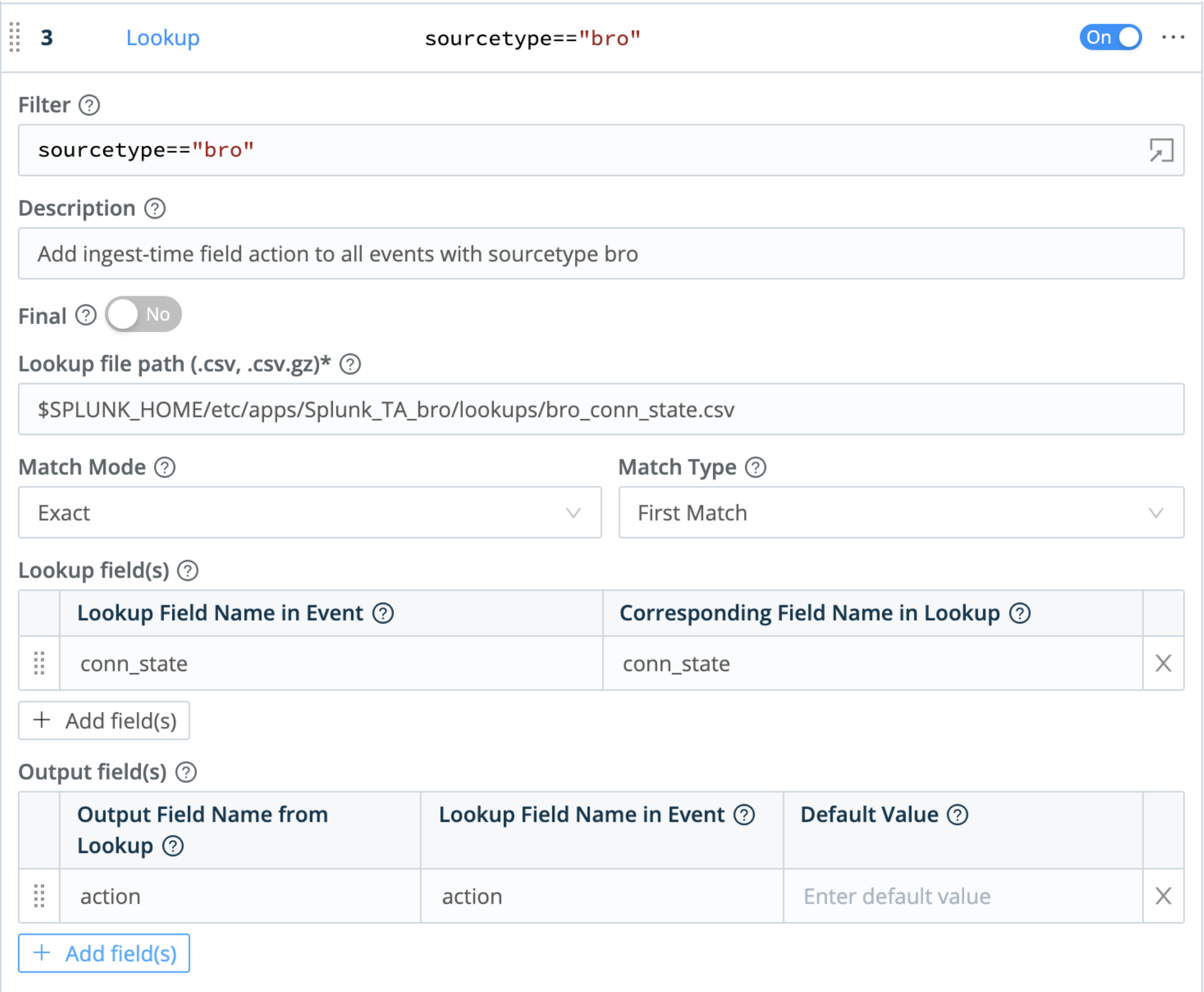
Let's assume we have the following lookup file. Given the field `conn_state` in an event, we would like to add a corresponding ingestion-time field called `action`.

```
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
reply from responder)."
allowed,S3,"Connection established and close attempt by responder seen (but no reply
from originator)."
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 from
the responder."
dropped,RSTRH,"Responder sent a SYN ACK followed by a RST, we never saw a SYN from
the (purported) originator."
dropped,SH,"Originator sent a SYN followed by a FIN, we never saw a SYN ACK from the
responder (hence the connection was 'half' open)."
dropped,SHR,"Responder sent a SYN ACK followed by a FIN, we never saw a SYN from the
originator."
allowed,OTH,"No SYN seen, just midstream traffic (a 'partial connection' that was
not later closed)."
```

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_bro/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`.



sourcetype=="bro"

Filter: sourcetype=="bro"

Description: Add ingest-time field action to all events with sourcetype bro

Final: No

Lookup file path (.csv, .csv.gz)*: $SPLUNK_HOME/etc/apps/Splunk_TA_bro/lookups/bro_conn_state.csv

Match Mode: Exact

Match Type: First Match

Lookup field(s)

Lookup Field Name in Event	Corresponding Field Name in Lookup
conn_state	conn_state

Output field(s)

Output Field Name from Lookup	Lookup Field Name in Event	Default Value
action	action	Enter default value

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

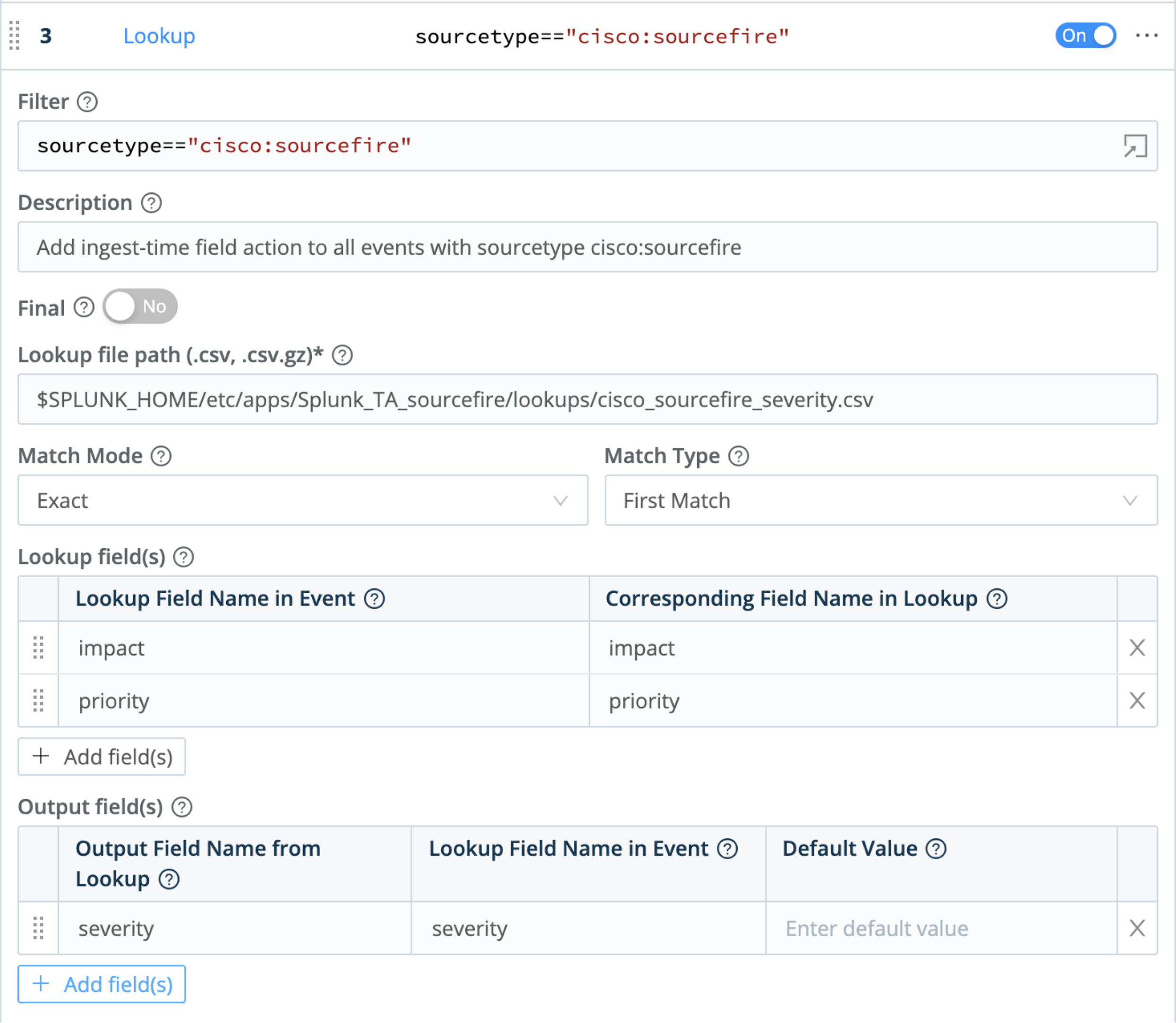
Let's assume we have the following lookup file, and given **both** the fields `impact` and `priority` in an event, we would like to add a corresponding ingestion-time field called `severity`.

```
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`.



3 Lookup `sourcetype=="cisco:sourcefire"` On

Filter: `sourcetype=="cisco:sourcefire"`

Description: Add ingest-time field action to all events with sourcetype cisco:sourcefire

Final: No

Lookup file path (.csv, .csv.gz)*: $SPLUNK_HOME/etc/apps/Splunk_TA_sourcefire/lookups/cisco_sourcefire_severity.csv

Match Mode: Exact

Match Type: First Match

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.

;

11.3.2. 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.

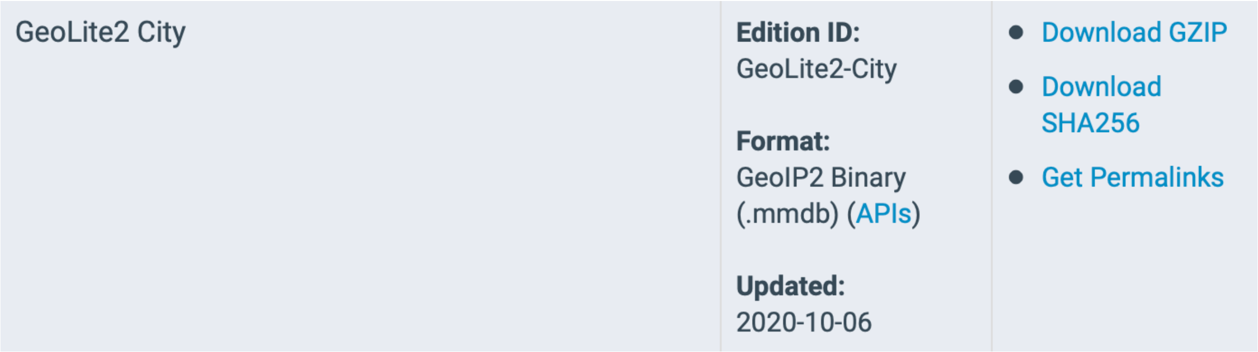
Download and Extract the Database

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, e.g.. placing it in the `$CRIBL_HOME/state` path. This will minimize the Git commit/deploy overhead around nightly updates to the binary database file.

Once in the database's directory, execute commands of this form:

```
$ scp *.mmdb <user>@<master-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

To set up automatic updates, see MaxMind's Automatic Updates for GeoIP2 and GeoIP Legacy Databases documentation. You'll need two modifications specific to Cribl Stream:

- This must be set up on the Leader, and on each Worker in any Group that uses GeoIP lookups.
- 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.

;

11.3.3. Lookups as Filters for Masks

Overview

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:

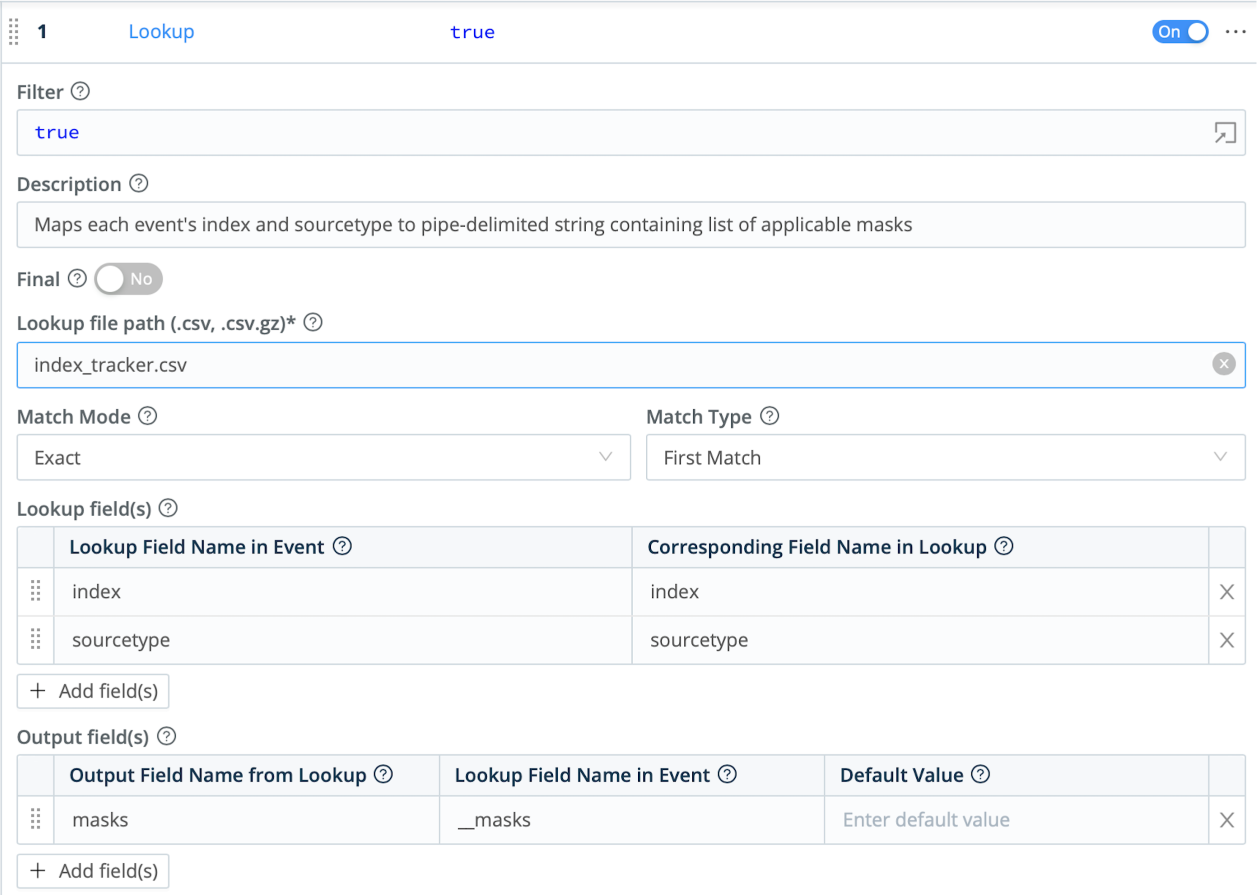
```
index,sourcetype,masks
apache_common, sourcetypec, ssn|credit_card|auth_token
syslog,sourcetypeb,ssn|auth_token
weblog,sourcetypea,auth_token|bearer_token
```

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:



1 Lookup true On

Filter: true

Description: Maps each event's index and sourcetype to pipe-delimited string containing list of applicable masks

Final: No

Lookup file path (.csv, .csv.gz)*: index_tracker.csv

Match Mode: Exact

Match Type: First Match

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

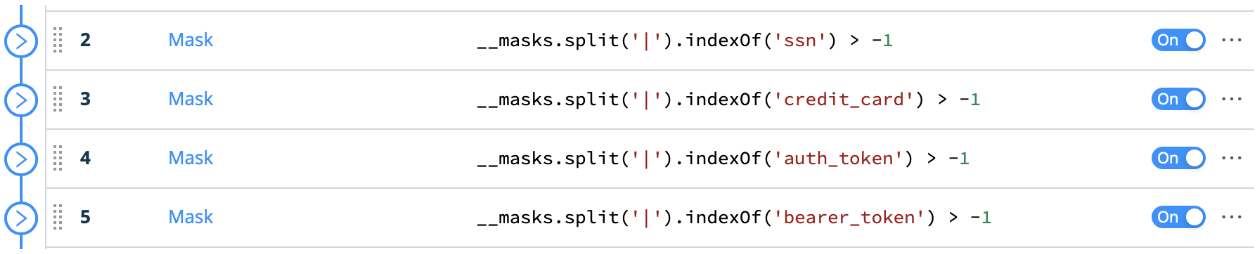
This Function keys against both the `index` and `sourcetype` fields. When it finds a matching combination, it adds a new key-value pair to your event for future filtering.

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:



2	Mask	`__masks.split('|').indexOf('ssn') > -1`	On
3	Mask	`__masks.split('|').indexOf('credit_card') > -1`	On
4	Mask	`__masks.split('|').indexOf('auth_token') > -1`	On
5	Mask	`__masks.split('|').indexOf('bearer_token') > -1`	On

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.

;

11.3.4. 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:

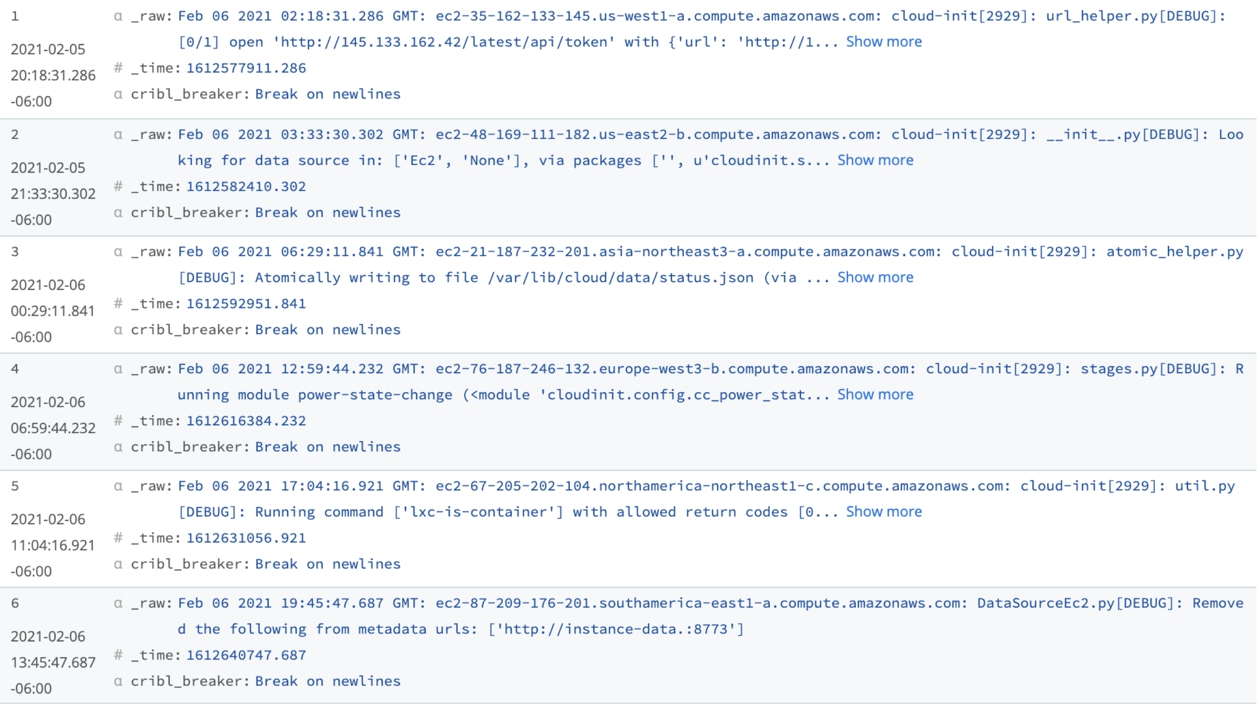
```
ec2-35-162-133-145.us-west1-a.compute.amazonaws.com
```

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:



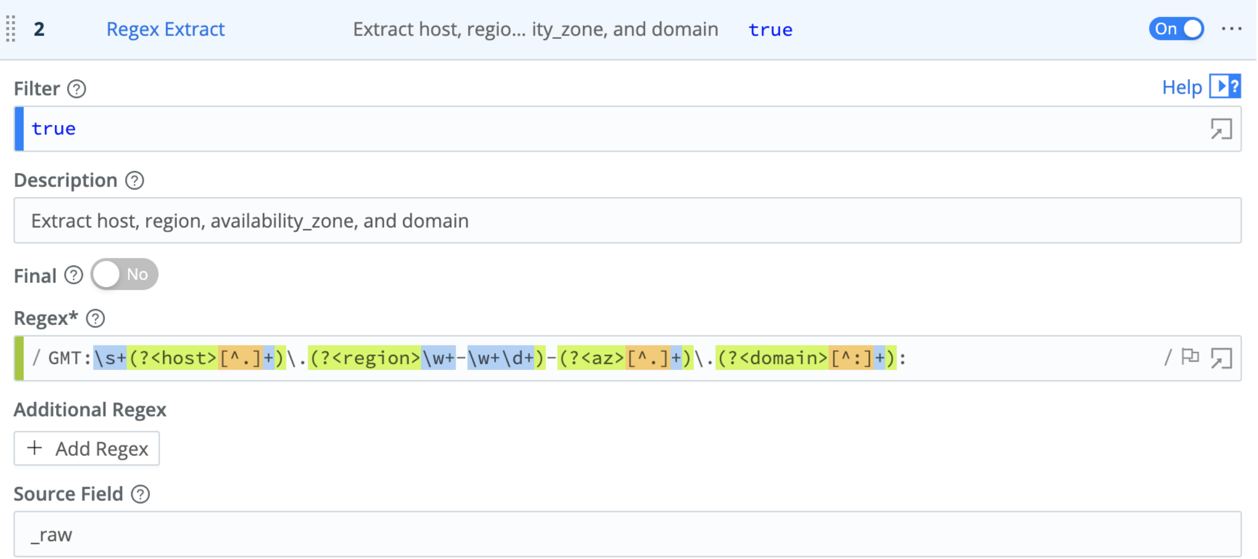
```
1           α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
2021-02-05          [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
20:18:31.286 # _time: 1612577911.286
-06:00      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:



2 Regex Extract Extract host, regio... ity_zone, and domain `true` On

Filter ⓘ

`true`

Description ⓘ

Extract host, region, availability_zone, and domain

Final ⓘ No

Regex* ⓘ

`/ GMT:\s+(?<host>[^.]+)\.(?<region>\w+-\w+\d+)-(?<az>[^.]+)\.(?<domain>[^:]+): /`

Additional Regex

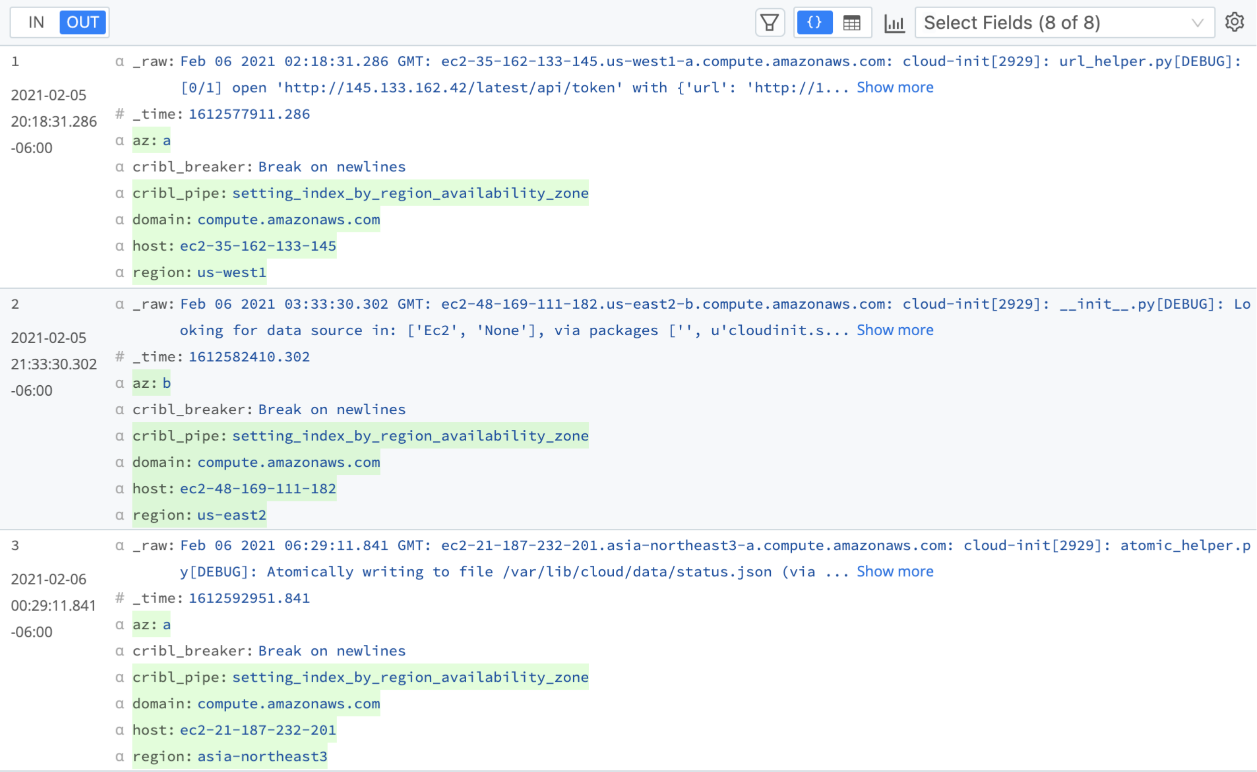
+ Add Regex

Source Field ⓘ

_raw

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

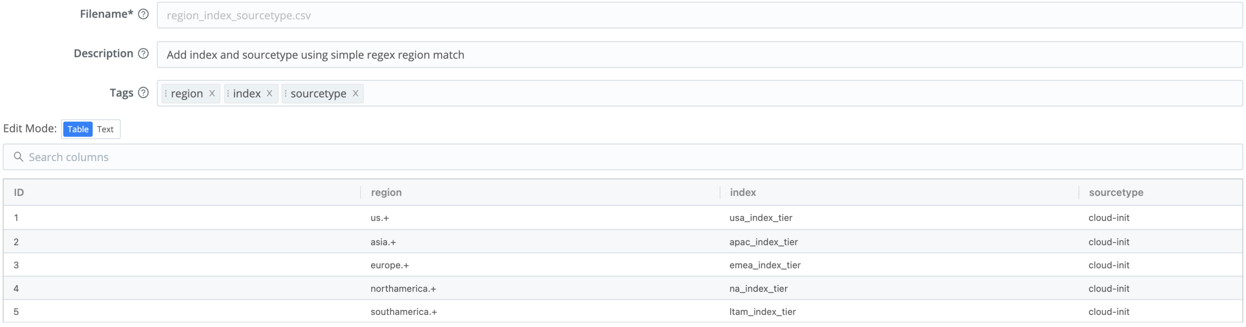
We still need to determine the index and sourcetype. Cribl Stream's Lookup Function enriches events with external fields. We'll use it with the newly extracted `region` field to assign an `index` and `sourcetype` to these events.

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 ×

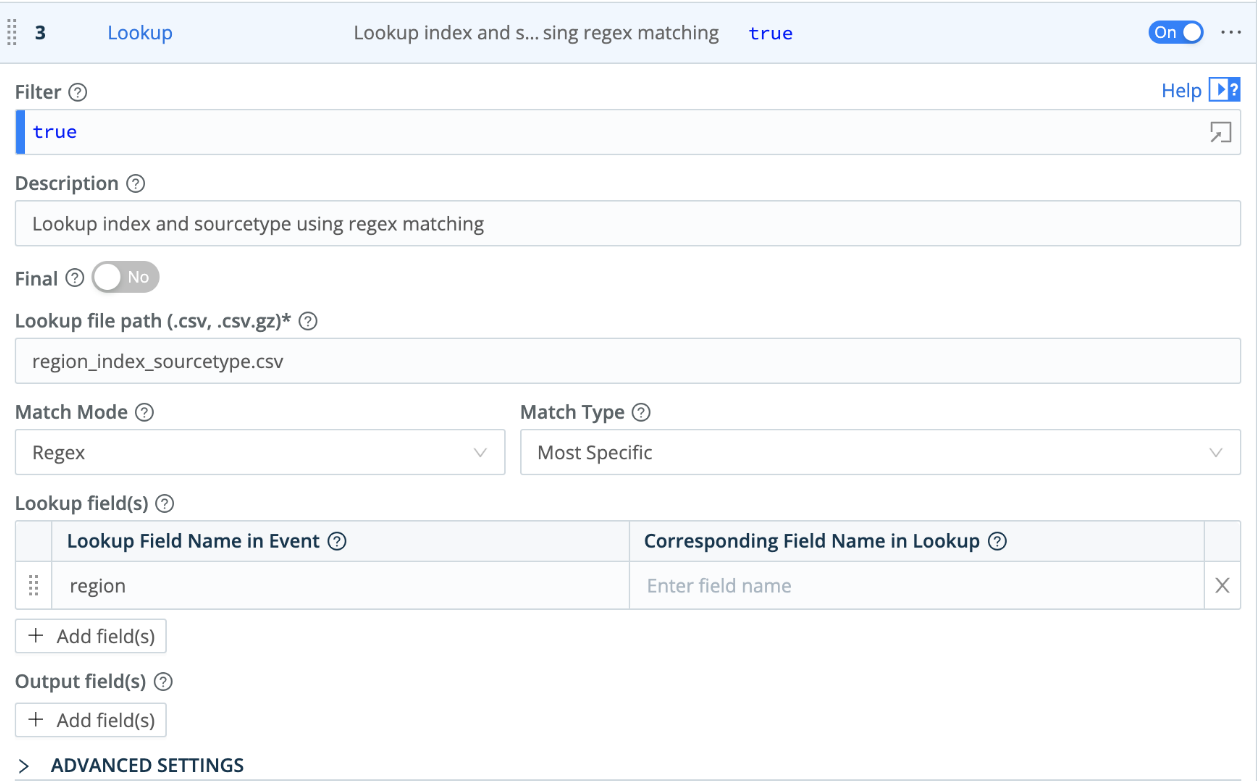
Edit Mode: Table Text

Search columns

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`.



3 Lookup Lookup index and s... sing regex matching `true` On

Filter ⓘ Help

`true`

Description ⓘ

Lookup index and sourcetype using regex matching

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

+ Add field(s)

Output field(s) ⓘ

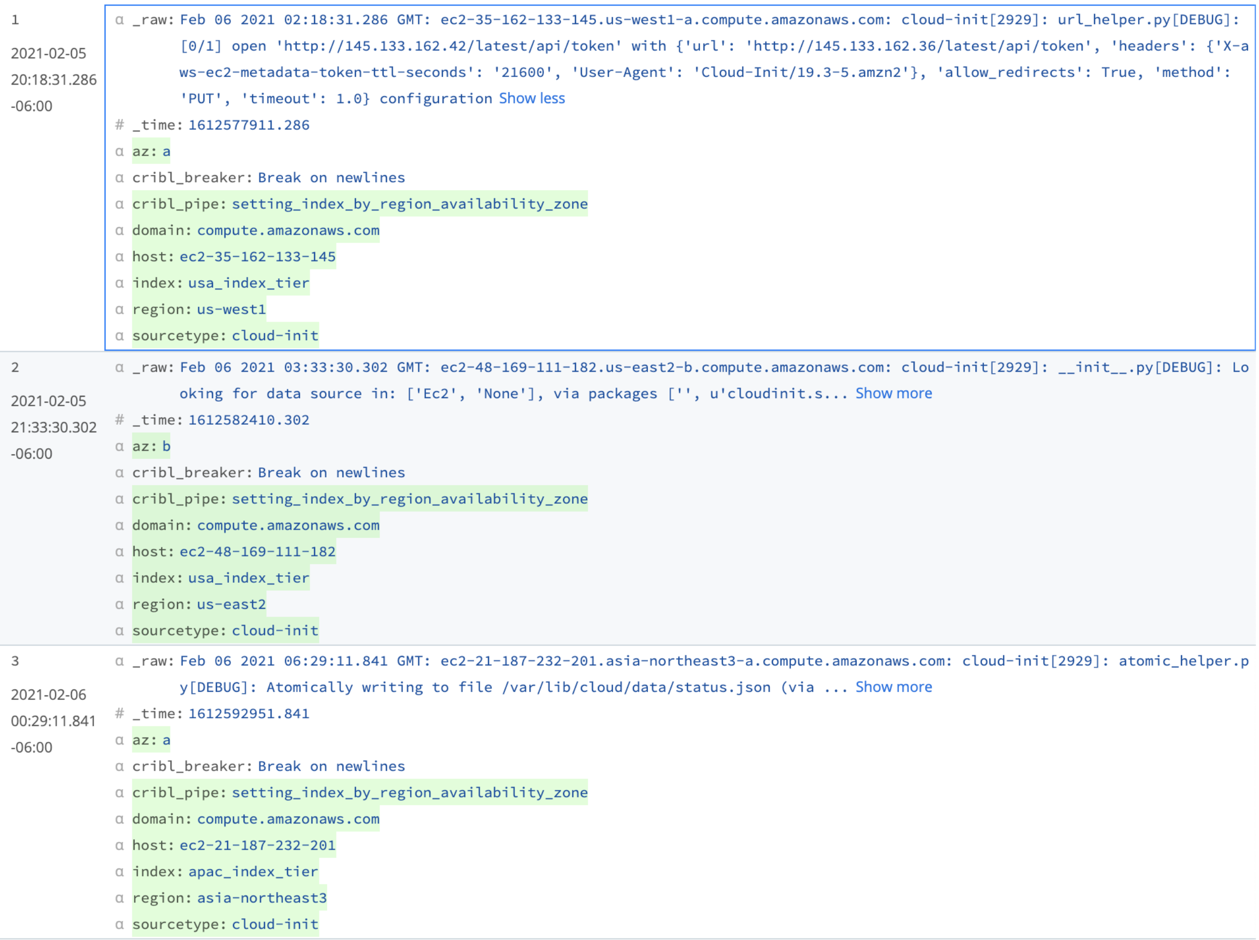
+ Add field(s)

> ADVANCED SETTINGS

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.



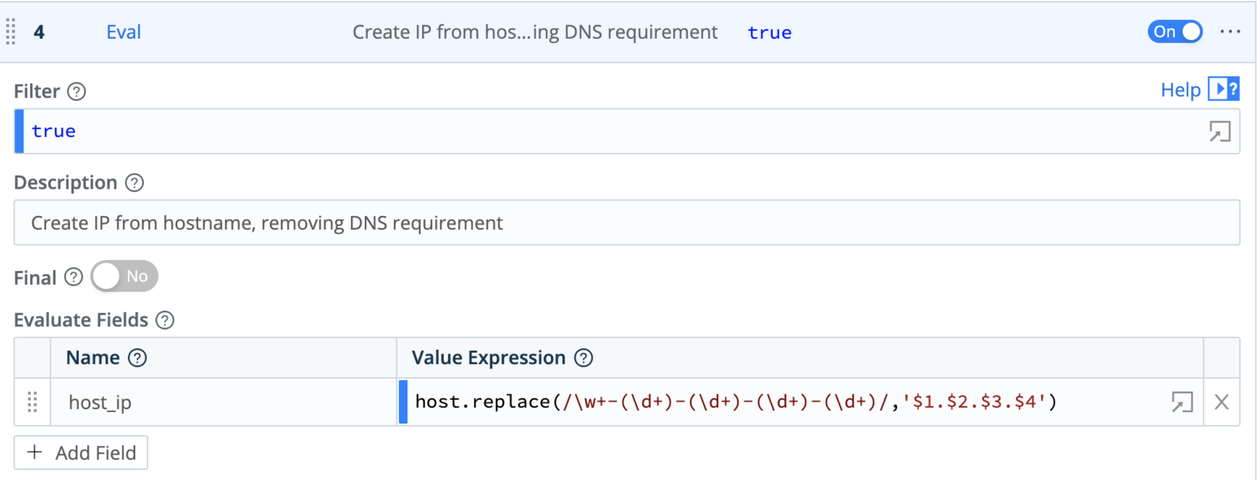
```
1            α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
2021-02-05          [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://145.133.162.36/latest/api/token', 'headers': {'X-a
20:18:31.286        ws-ec2-metadata-token-ttl-seconds': '21600', 'User-Agent': 'Cloud-Init/19.3-5.amzn2'}, 'allow_redirects': True, 'method':
-06:00              'PUT', 'timeout': 1.0} configuration Show less
             # _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
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       α az: b
             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
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       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: ec2-21-187-232-201
             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!



4 Eval Create IP from hos...ing DNS requirement `true` On

Filter: `true`

Description: Create IP from hostname, removing DNS requirement

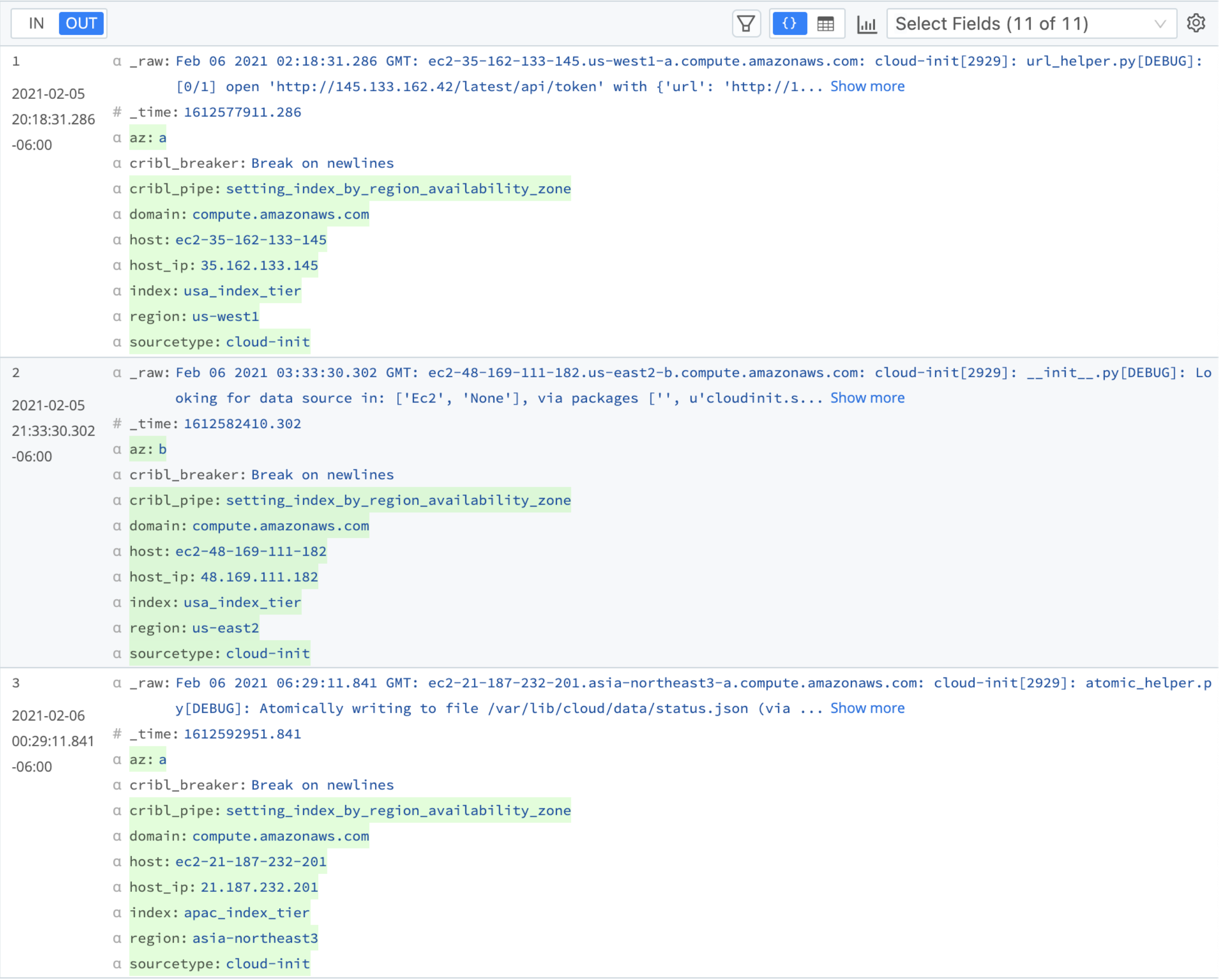
Final: No

Evaluate Fields

Name	Value Expression
host_ip	`host.replace(/\w+-(\d+)-(\d+)-(\d+)-(\d+)/,'$1.$2.$3.$4')`

Results of the Eval Function and Replace Method

The `host_ip` field is now added to the events, displayed below `host`:

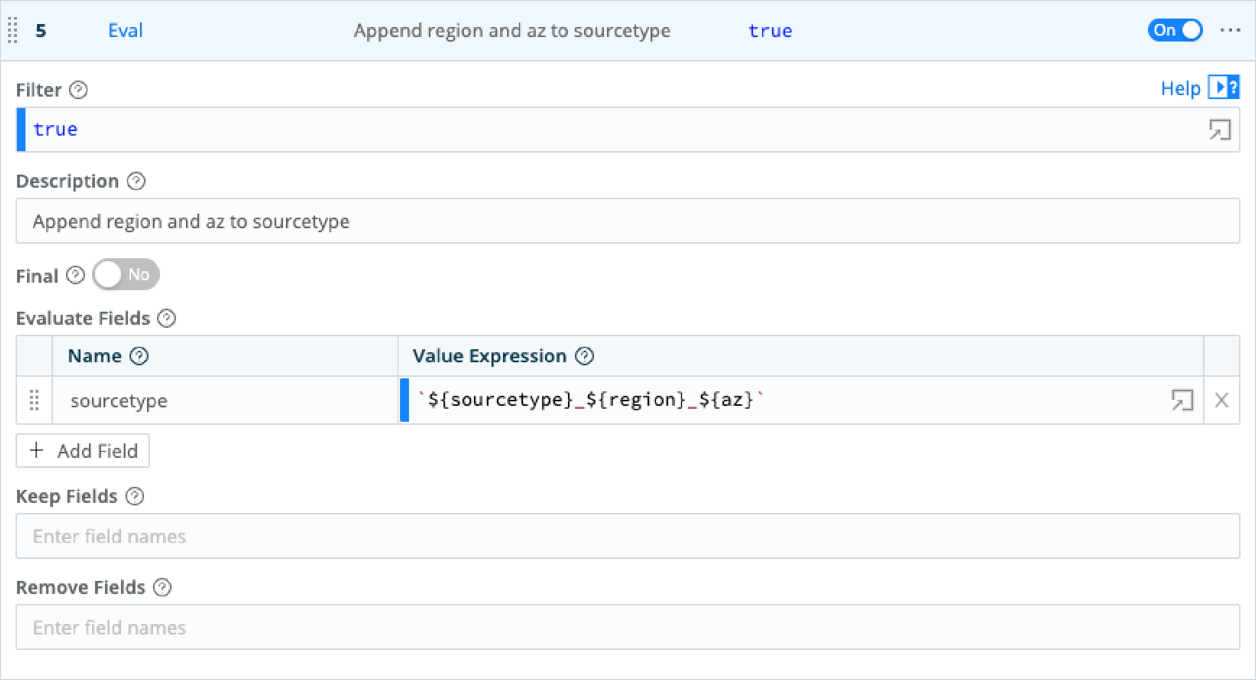


```
1
2021-02-05 20:18:31.286 -06:00
α _raw: Feb 06 2021 02:18:31.286 GMT: ec2-35-162-133-145.us-west1-a.compute.amazonaws.com: cloud-init[2929]: url_helper.py[DEBUG]:
        [0/1] open 'http://145.133.162.42/latest/api/token' with {'url': 'http://1... Show more
# _time: 1612577911.286
α az: a
α cribl_breaker: 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 host_ip: 35.162.133.145
α index: usa_index_tier
α region: us-west1
α sourcetype: cloud-init

2
2021-02-05 21:33:30.302 -06:00
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
# _time: 1612582410.302
α az: b
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 host_ip: 48.169.111.182
α index: usa_index_tier
α region: us-east2
α sourcetype: cloud-init

3
2021-02-06 00:29:11.841 -06:00
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
# _time: 1612592951.841
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: ec2-21-187-232-201
α host_ip: 21.187.232.201
α index: apac_index_tier
α region: asia-northeast3
α 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.



5 Eval Append region and az to sourcetype true On

Filter: `true`

Description: Append region and az to sourcetype

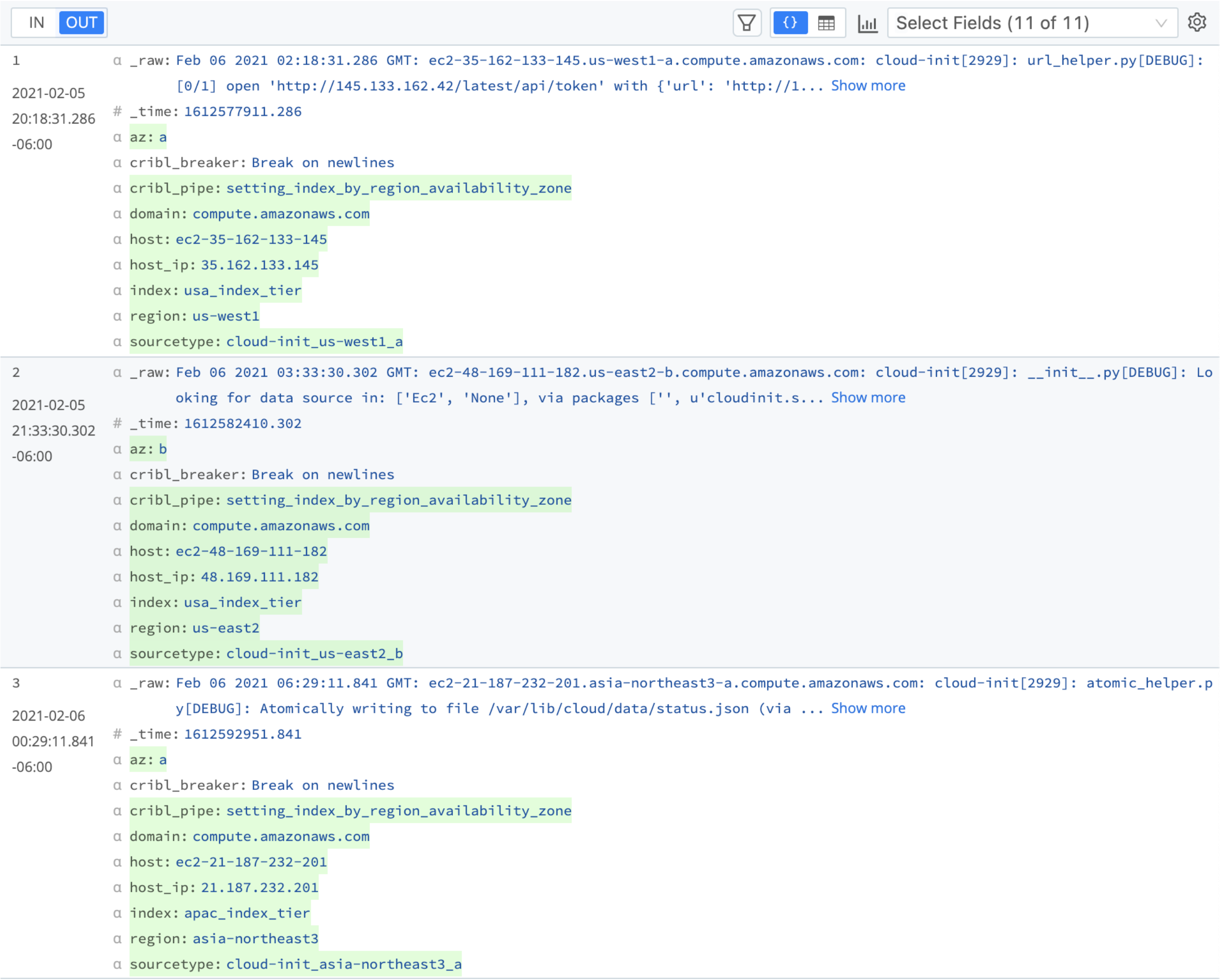
Final: No

Evaluate Fields

Name	Value Expression
sourcetype	`` `${sourcetype}_${region}_${az}` ``

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           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 host_ip: 35.162.133.145
            α index: usa_index_tier
            α region: us-west1
            α sourcetype: cloud-init_us-west1_a
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 host_ip: 48.169.111.182
            α index: usa_index_tier
            α region: us-east2
            α sourcetype: cloud-init_us-east2_b
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 host_ip: 21.187.232.201
            α index: apac_index_tier
            α region: asia-northeast3
            α 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 **+ Add New** 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
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
regular expressions in a Cribl Stream Pipeline including field extraction, lookup,
replace, and field value manipulation."
        },
        "id": "comment"
      },
      {
        "filter": "true",
        "conf": {
          "source": "_raw",
          "iterations": 100,
          "overwrite": false,
          "regex": "/GMT:\\s+(?<host>[^.]+)\\.(?<region>\\w+-\\w+\\d+)-(?<az>
[^.]+)\\.(?<domain>[^:]+):/"
        },
        "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.$4')"
            }
          ]
        },
        "id": "eval",
        "description": "Create IP from hostname, removing DNS requirement",
        "disabled": false
      },
      {
        "filter": "true",
        "conf": {
          "add": [
            {
              "name": "sourcetype",
              "value": "`${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:

```
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-aws-ec2-metadata-token-ttl-
seconds': '21600', 'User-Agent': 'Cloud-Init/19.3-5.amzn2'}, 'allow_redirects':
True, 'method': 'PUT', 'timeout': 1.0} configuration
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]: Looking for data source in: ['Ec2', 'None'],
via packages ['', u'cloudinit.sources'] that matches dependencies ['FILESYSTEM']
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[DEBUG]: Atomically
writing to file /var/lib/cloud/data/status.json (via temporary file
/var/lib/cloud/data/tmpS7ibzJ) - w:[644] 489 bytes/chars
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]: Running module power-
state-change (<module 'cloudinit.config.cc_power_state_change' from
'/usr/lib/python2.7/site-packages/cloudinit/config/cc_power_state_change.pyc'>) with
frequency once-per-instance
Feb 06 2021 17:04:16.921 GMT: ec2-67-205-202-104.northamerica-northeast1-
c.compute.amazonaws.com: cloud-init[2929]: util.py[DEBUG]: Running command ['lxc-is-
container'] with allowed return codes [0] (shell=False, capture=True)
Feb 06 2021 19:45:47.687 GMT: ec2-87-209-176-201.southamerica-east1-
a.compute.amazonaws.com: DataSourceEc2.py[DEBUG]: Removed the following from
metadata urls: ['http://instance-data.:8773']
```

From here, modify the sample data, lookup table, and Functions to adapt this approach to your own needs!

;

11.4. Sampling Applications

11.4.1. 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.

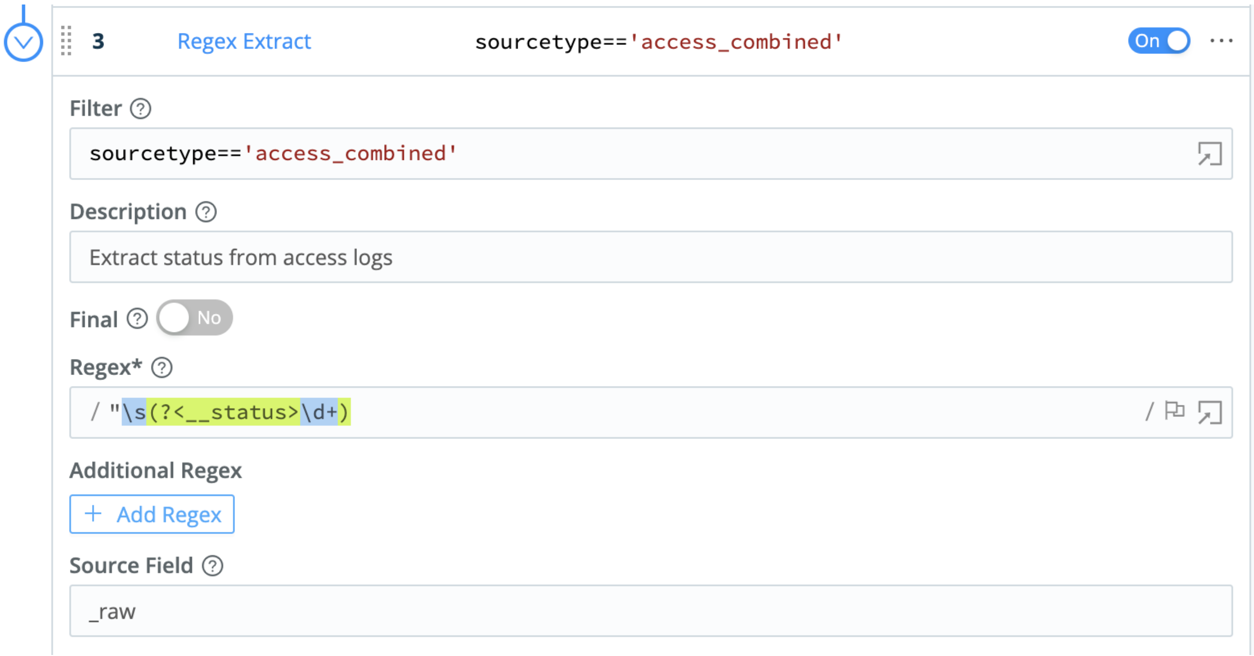
See the example below, or see more details in Access Logs and Firewall Logs.

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.



3 Regex Extract sourcetype=='access_combined' On

Filter

```
sourcetype=='access_combined'
```

Description

Extract status from access logs

Final No

Regex*

```
/ "\s(?<__status>\d+)
```

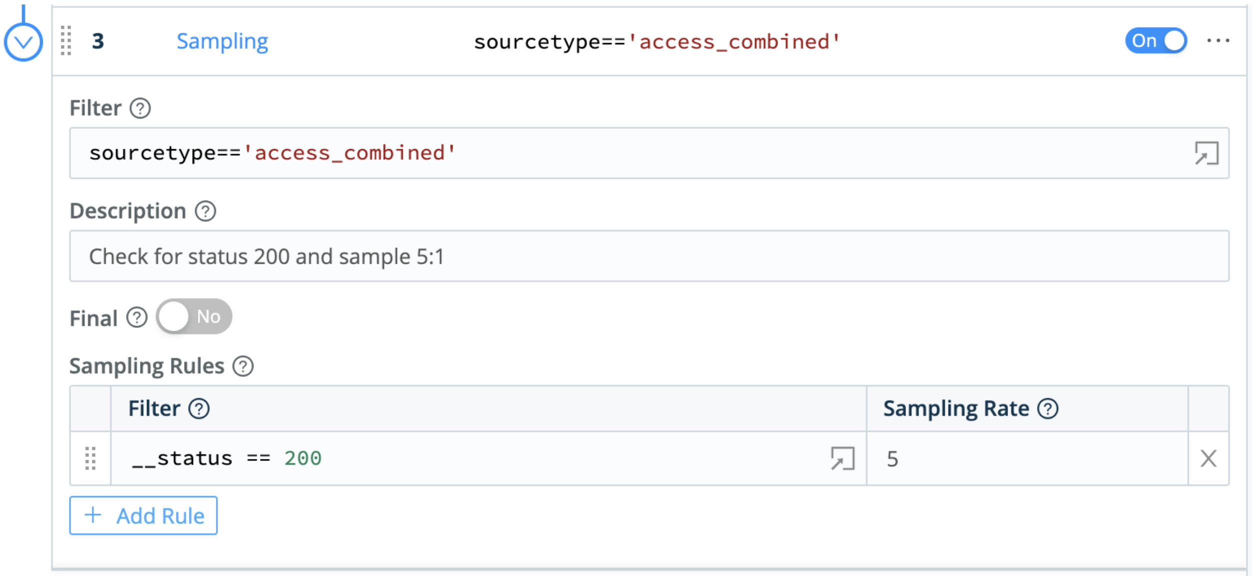
Additional Regex

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`.



3 Sampling sourcetype=='access_combined' On

Filter

```
sourcetype=='access_combined'
```

Description

Check for status 200 and sample 5:1

Final No

Sampling Rules

Filter	Sampling Rate
`__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.

;

11.4.2. 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.

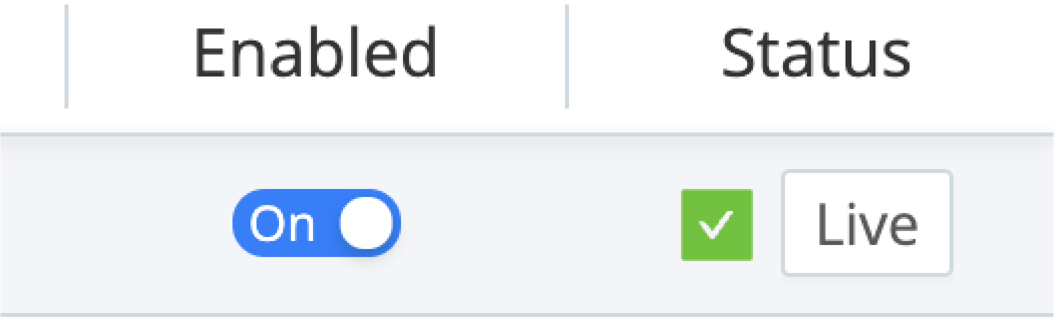
Configure the Source you need. For simplicity, use port 1024 or higher. (If you must use lower ports you'll need to follow a different installation procedure.)

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.

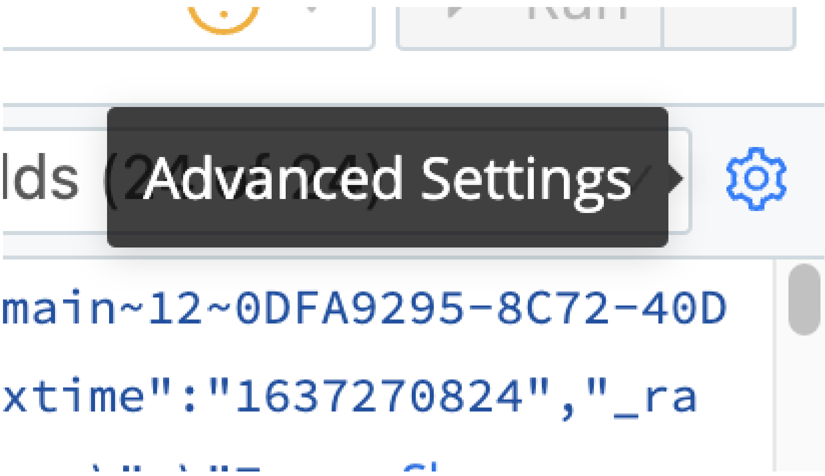
When finished, click **Save as Sample File**. Note the file name (and rename if you like). Close the sample window.

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**:



```
main~12~0DFA9295-8C72-40D
xtime":"1637270824","_ra
```

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.

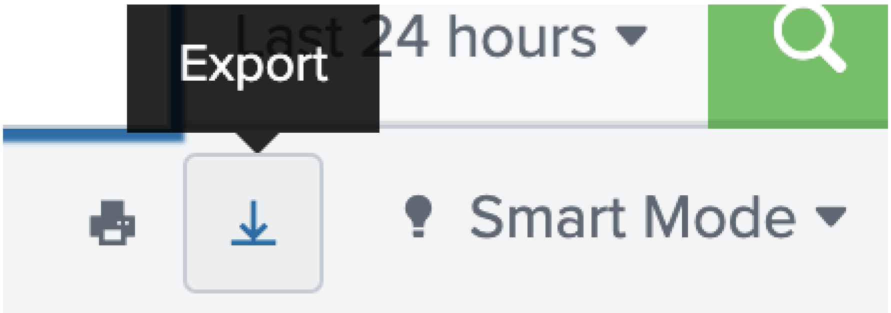
In the Splunk GUI:

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
```

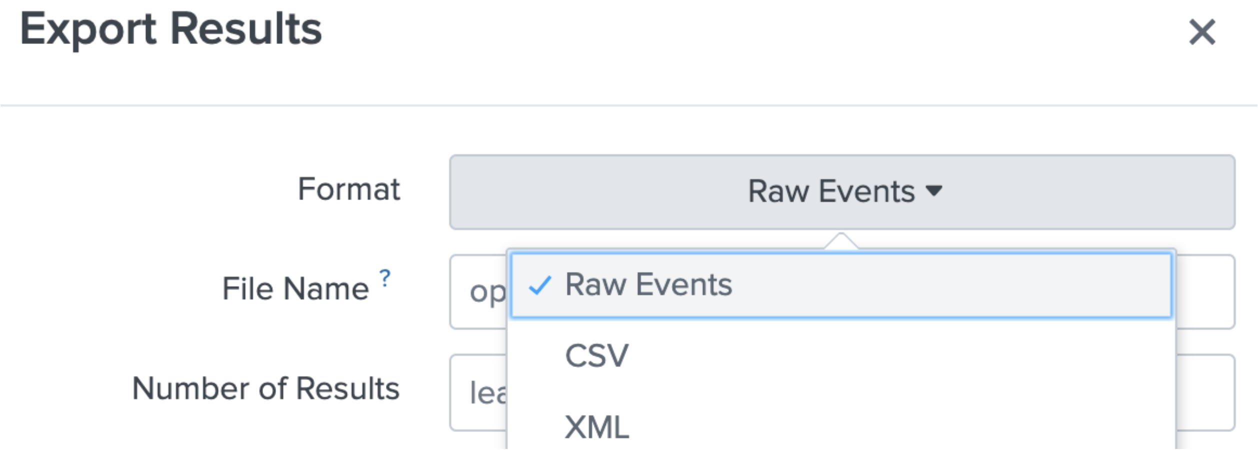
2. Adjust the search as needed to exclude or include the events you want to send.

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

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" -output rawdata > cisco_asa_sample.txt
```

;

11.4.3. 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.

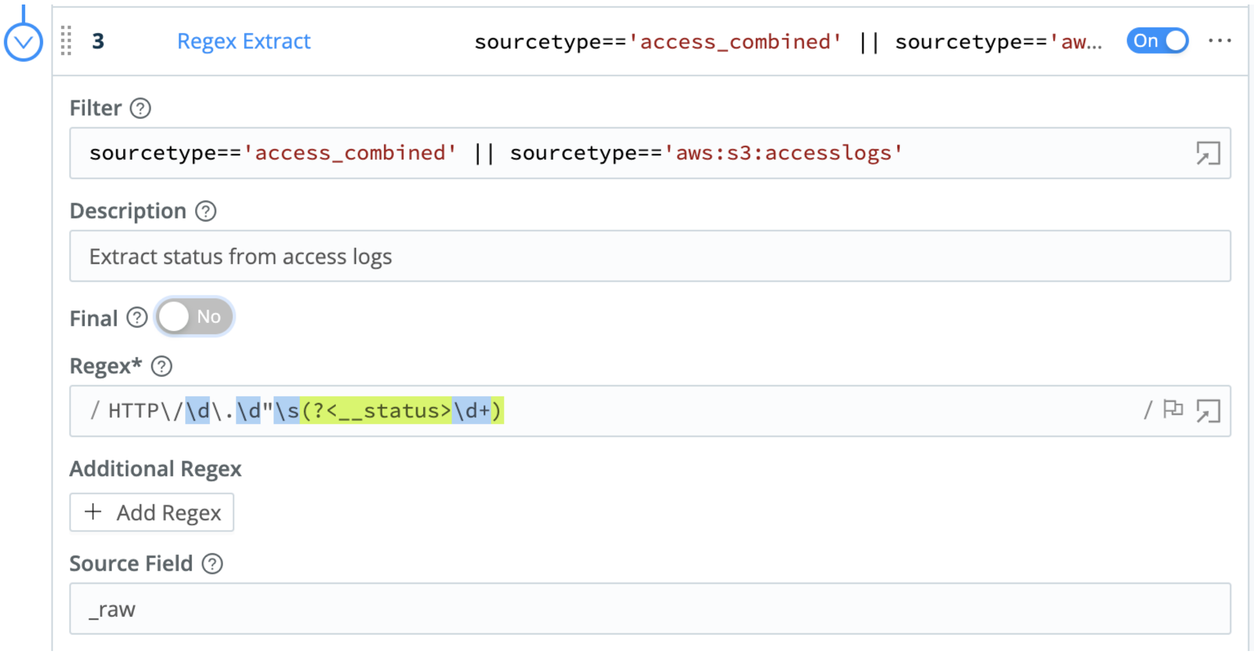
Using `status` as the Sampling Condition

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+)/`



Filter: `sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'`

Description: Extract status from access logs

Final: No

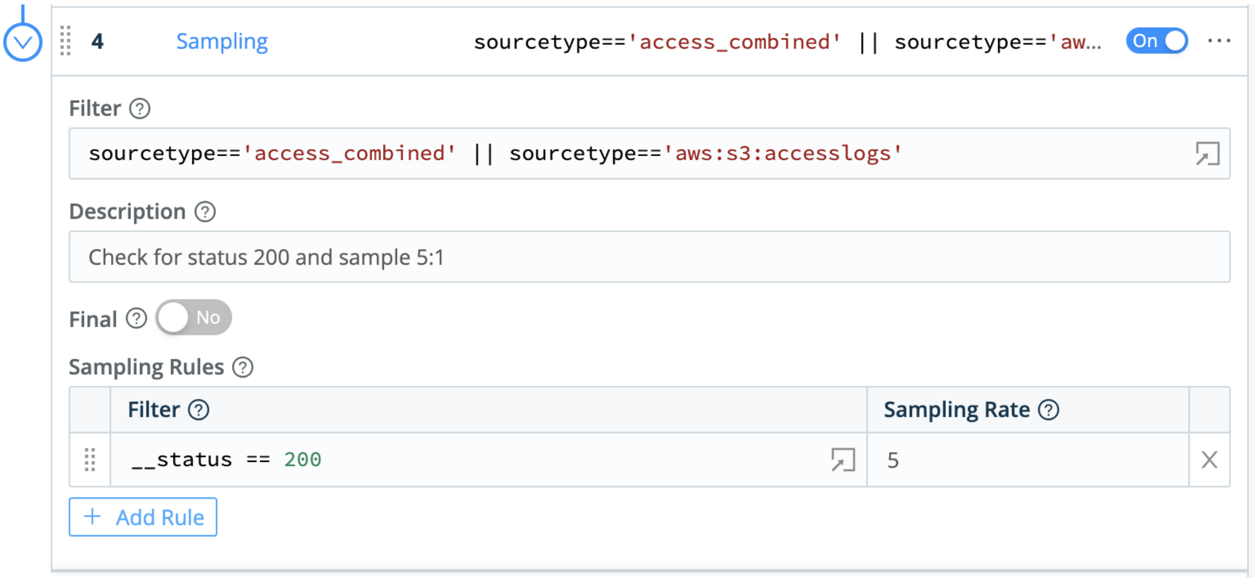
Regex*: `/ HTTP\/\d\.\d"\s(?<__status>\d+) /`

Source Field: _raw

Defining the Regex Extract Function

3. Add a Sampling Function to sample `5:1` when `__status==200`.

4. Save.



Filter: `sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'`

Description: Check for status 200 and sample 5:1

Final: No

Filter	Sampling Rate
`__status == 200`	5

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'`

;

11.4.4. 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.

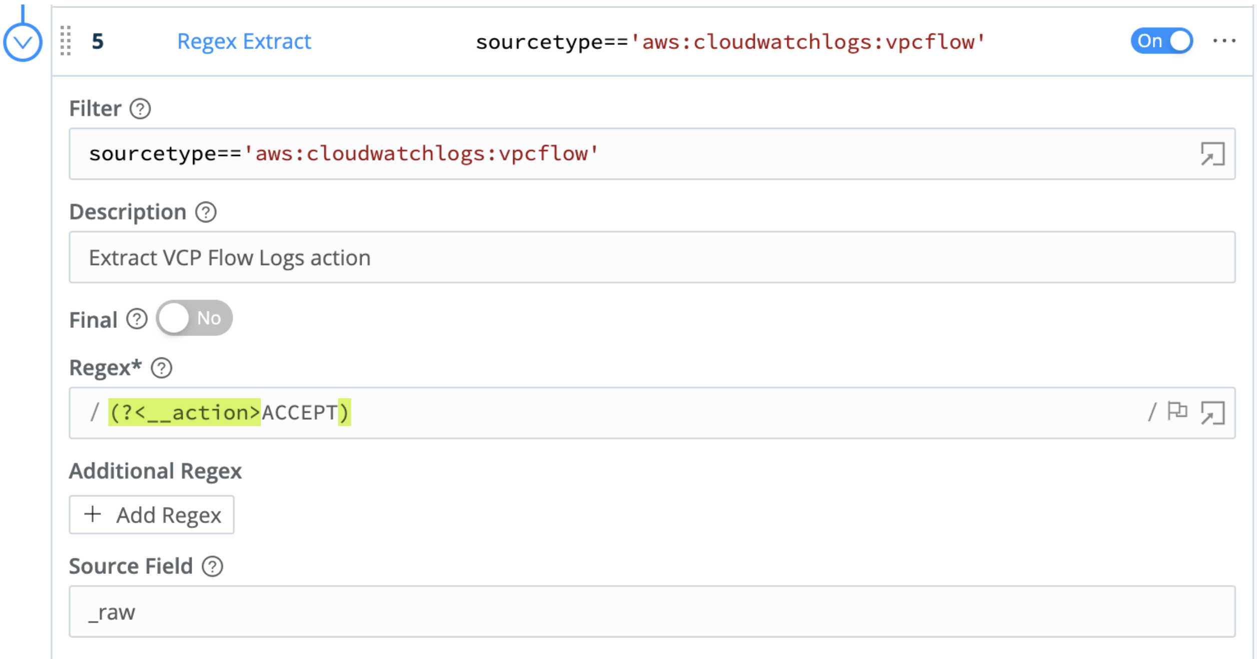
Typical VPC Flow Logs look like this:

```
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
```

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)/`



5 Regex Extract sourcetype=='aws:cloudwatchlogs:vpcflow' On

Filter
sourcetype=='aws:cloudwatchlogs:vpcflow'

Description
Extract VCP Flow Logs action

Final No

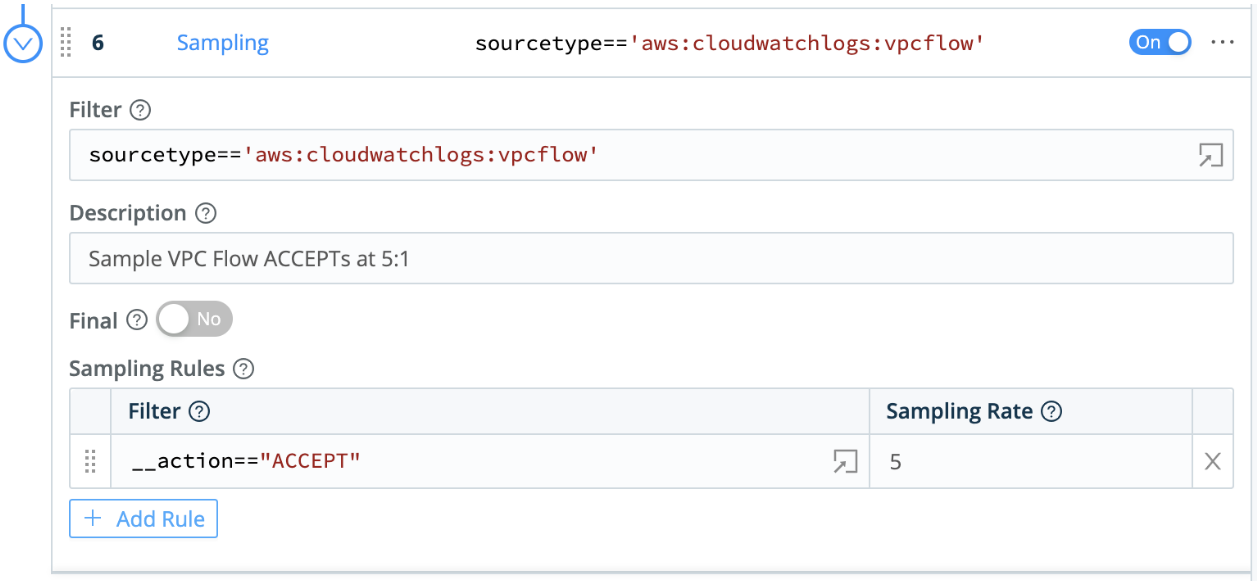
Regex*
/ (?<__action>ACCEPT) /

Additional Regex
+ Add Regex

Source Field
_raw

Extracting the `__action` field

3. Add a **Sampling** Function to sample `5:1` when `__action=="ACCEPT"`.
4. Save.



6 Sampling sourcetype=='aws:cloudwatchlogs:vpcflow' On

Filter
sourcetype=='aws:cloudwatchlogs:vpcflow'

Description
Sample VPC Flow ACCEPTs at 5:1

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'`

;

11.5. Using Other Functions

11.5.1. Code Function Examples

The Code Function introduced in Cribl Stream version 3.1 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

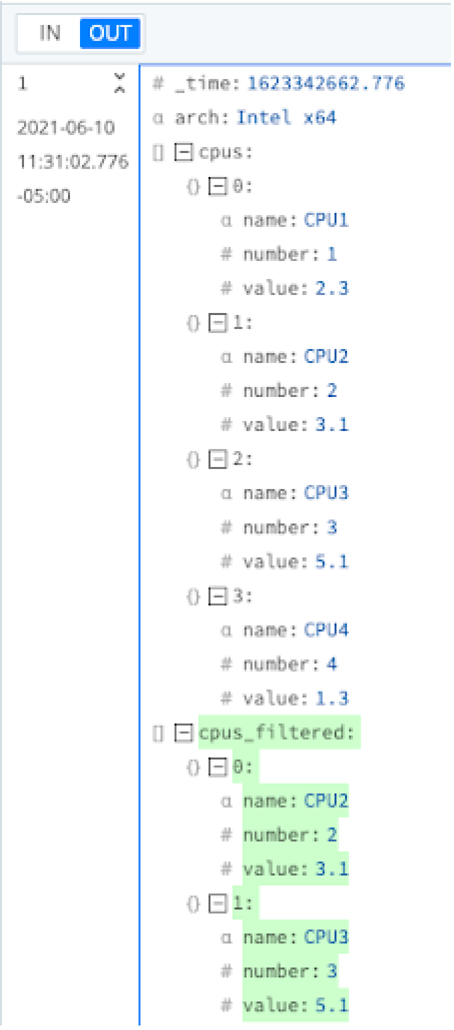
To create a new field, it is as simple as assigning the field to a value. For example, if you want to create a field `test` with the value `Hello, Goats!`, use the following syntax:

```
__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)
```



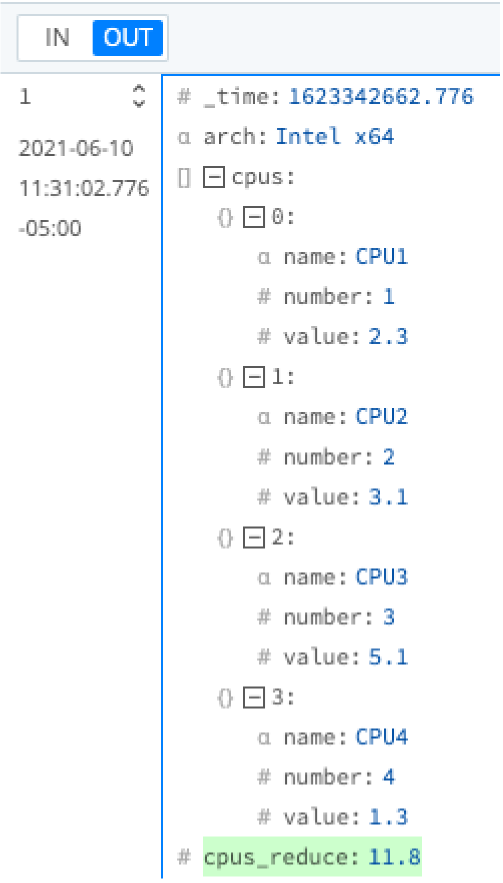
```
1     2021-06-10
      11:31:02.776
      -05:00

# _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.value, 0)
```



```
1                  # _time: 1623342662.776
                   α arch: Intel x64
2021-06-10         [] ⊟ cpus:
11:31:02.776           {} ⊟ 0:
-05:00                     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_reduce: 11.8
```

Code Function example: Reduce

Some/Every

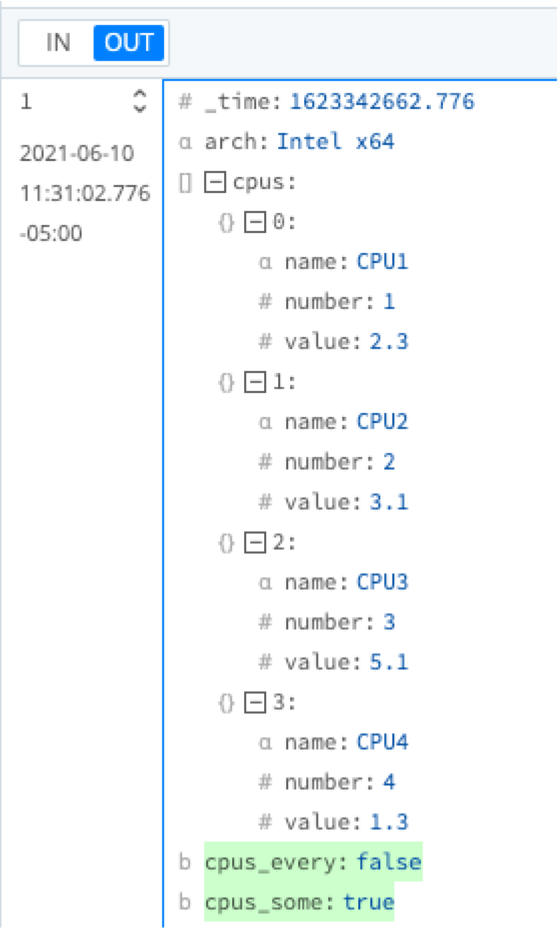
The `some` and `every` methods return a boolean result (`true` / `false`).

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)
```



```
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
b cpus_every: false
b cpus_some: true
```

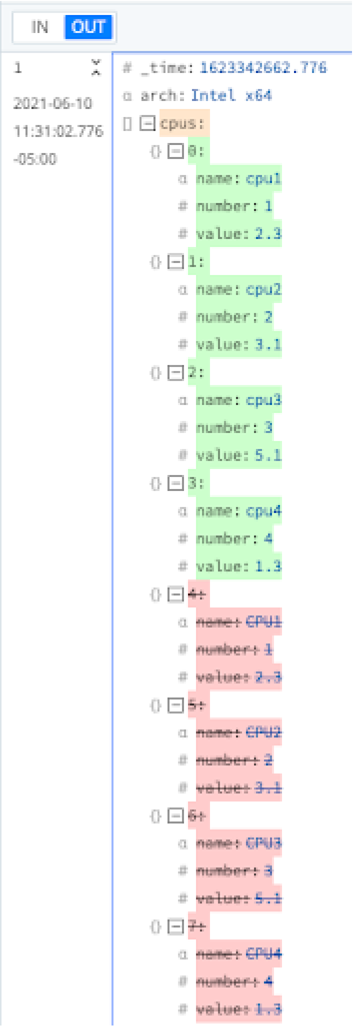
Code Function example: Some/Every

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.toLowerCase()}))
```



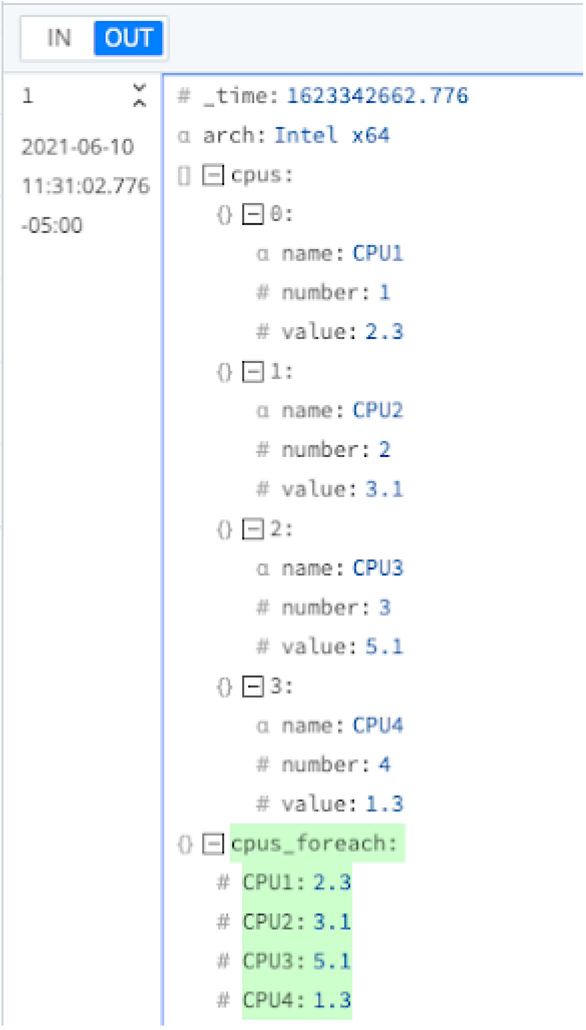
```
1
2021-06-10
11:31:02.776
-05:00

_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            # _time: 1623342662.776
2021-06-10   a arch: Intel x64
11:31:02.776 [] ⊟ cpus:
-05:00            {} ⊟ 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

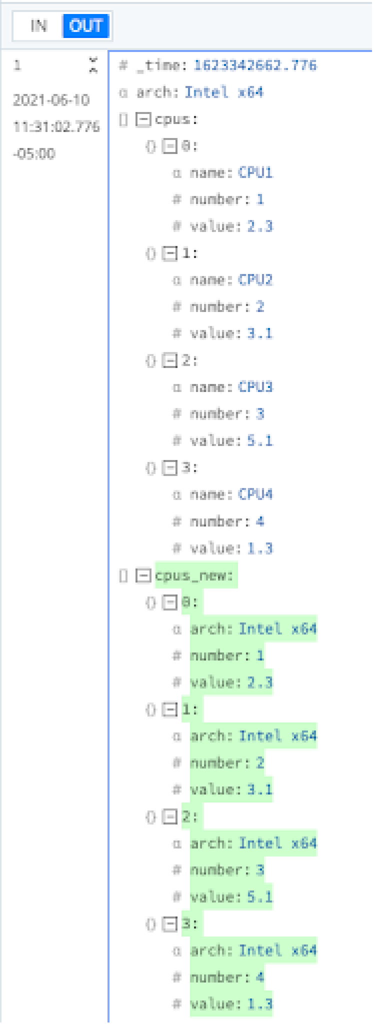
You could also accomplish the same result by replacing the middle line above with the `map` method below:

```
__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;
})
```



```
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
```

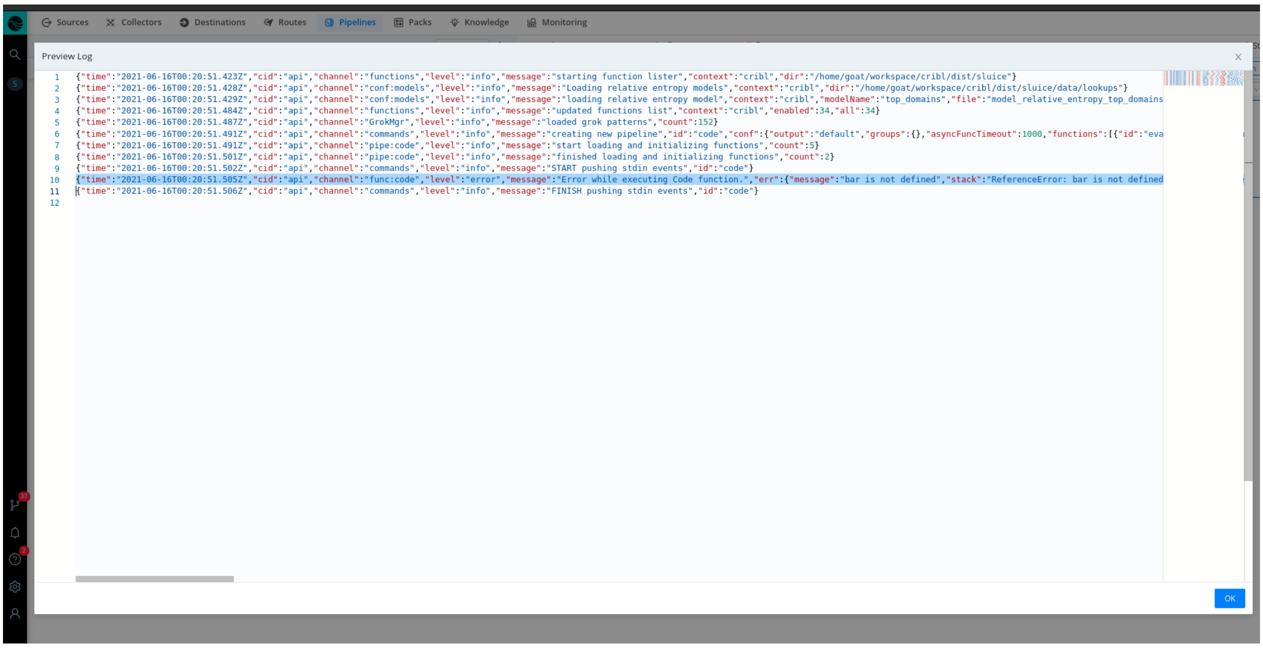
Code Function example: Build a new array

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**.

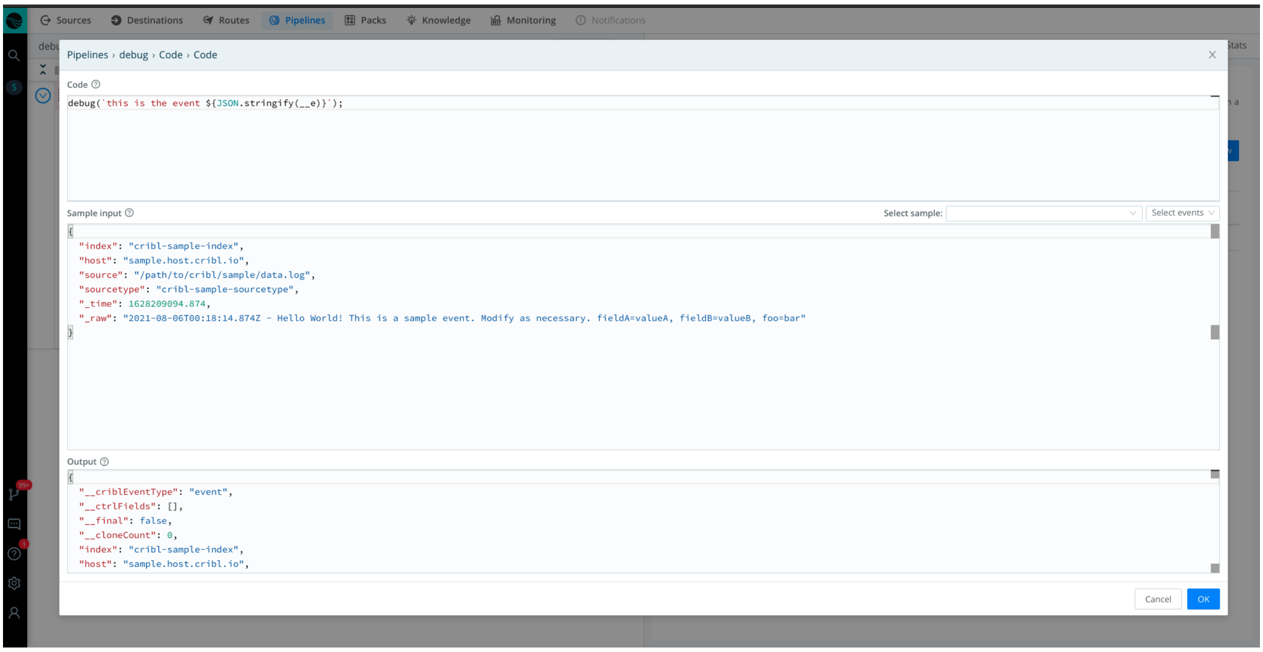


```
{"time":"2021-06-16T00:20:51.423Z","cid":"api","channel":"functions","level":"info","message":"starting function lister","context":"cribl","dir":"/home/goat/workspace/cribl/dist/sluice"}
{"time":"2021-06-16T00:20:51.428Z","cid":"api","channel":"conf:models","level":"info","message":"Loading relative entropy models","context":"cribl","dir":"/home/goat/workspace/cribl/dist/sluice/data/lookups"}
{"time":"2021-06-16T00:20:51.429Z","cid":"api","channel":"conf:models","level":"info","message":"loading relative entropy model","context":"cribl","modelName":"top_domains","file":"model_relative_entropy_top_domains
{"time":"2021-06-16T00:20:51.484Z","cid":"api","channel":"functions","level":"info","message":"updated functions list","context":"cribl","enabled":34,"all":34}
{"time":"2021-06-16T00:20:51.487Z","cid":"api","channel":"GrokMgr","level":"info","message":"loaded grok patterns","count":152}
{"time":"2021-06-16T00:20:51.491Z","cid":"api","channel":"commands","level":"info","message":"creating new pipeline","id":"code","conf":{"output":"default","groups":{},"asyncFuncTimeout":1000,"functions":[{"id":"eva
{"time":"2021-06-16T00:20:51.491Z","cid":"api","channel":"pipe:code","level":"info","message":"start loading and initializing functions","count":5}
{"time":"2021-06-16T00:20:51.501Z","cid":"api","channel":"pipe:code","level":"info","message":"finished loading and initializing functions","count":2}
{"time":"2021-06-16T00:20:51.502Z","cid":"api","channel":"commands","level":"info","message":"START pushing stdin events","id":"code"}
{"time":"2021-06-16T00:20:51.505Z","cid":"api","channel":"func:code","level":"error","message":"Error while executing Code function.","err":{"message":"bar is not defined","stack":"ReferenceError: bar is not defined
{"time":"2021-06-16T00:20:51.506Z","cid":"api","channel":"commands","level":"info","message":"FINISH pushing stdin events","id":"code"}
```

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.



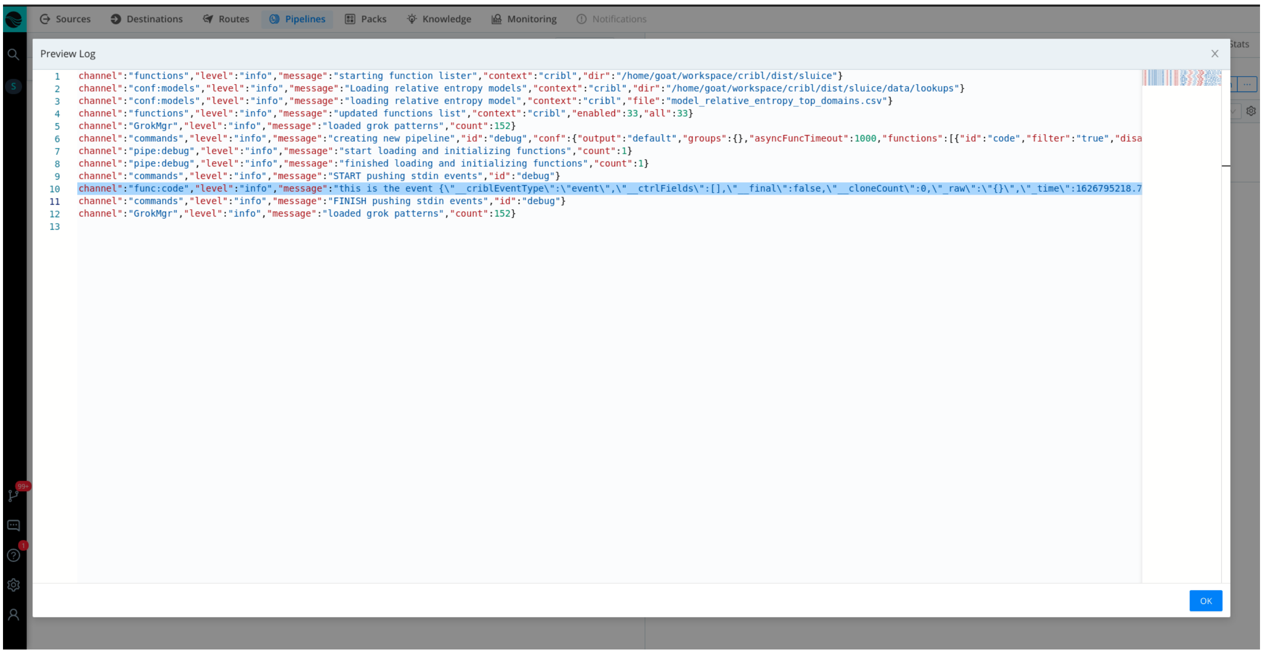
```
debug(`this is the event ${JSON.stringify(__e)}`);
```

```
{
  "index": "cribl-sample-index",
  "host": "sample.host.cribl.io",
  "source": "/path/to/cribl/sample/data.log",
  "sourcetype": "cribl-sample-sourcetype",
  "_time": 1628209094.874,
  "_raw": "2021-08-06T00:18:14.874Z - Hello World! This is a sample event. Modify as necessary. fieldA=valueA, fieldB=valueB, foo=bar"
}
```

```
{
  "__criblEventType": "event",
  "__ctrlFields": [],
  "__final": false,
  "__cloneCount": 0,
  "index": "cribl-sample-index",
  "host": "sample.host.cribl.io",
```

Debugging a Code Function

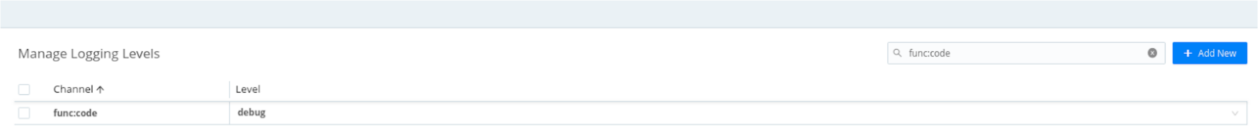
Messages logged by this `debug` helper function are shown in the Preview Log by default.



```
channel":"functions","level":"info","message":"starting function lister","context":"cribl","dir":"/home/goat/workspace/cribl/dist/sluice"}
channel":"conf:models","level":"info","message":"Loading relative entropy models","context":"cribl","dir":"/home/goat/workspace/cribl/dist/sluice/data/lookups"}
channel":"conf:models","level":"info","message":"loading relative entropy model","context":"cribl","file":"model_relative_entropy_top_domains.csv"}
channel":"functions","level":"info","message":"updated functions list","context":"cribl","enabled":33,"all":33}
channel":"GrokMgr","level":"info","message":"loaded grok patterns","count":152}
channel":"commands","level":"info","message":"creating new pipeline","id":"debug","conf":{"output":"default","groups":{},"asyncFuncTimeout":1000,"functions":[{"id":"code","filter":"true","disa
channel":"pipe:debug","level":"info","message":"start loading and initializing functions","count":1}
channel":"pipe:debug","level":"info","message":"finished loading and initializing functions","count":1}
channel":"commands","level":"info","message":"START pushing stdin events","id":"debug"}
channel":"func:code","level":"info","message":"this is the event {\"__criblEventType\":\"event\",\"__ctrlFields\":[],\"__final\":false,\"__cloneCount\":0,\"_raw\":\"{}\",\"_time\":1626795218.7
channel":"commands","level":"info","message":"FINISH pushing stdin events","id":"debug"}
channel":"GrokMgr","level":"info","message":"loaded grok patterns","count":152}
```

Debugging a Code Function with the Preview Log

You can also route these messages to regular logs, although this requires setting the Function's log level (`func:code`) to `debug`.

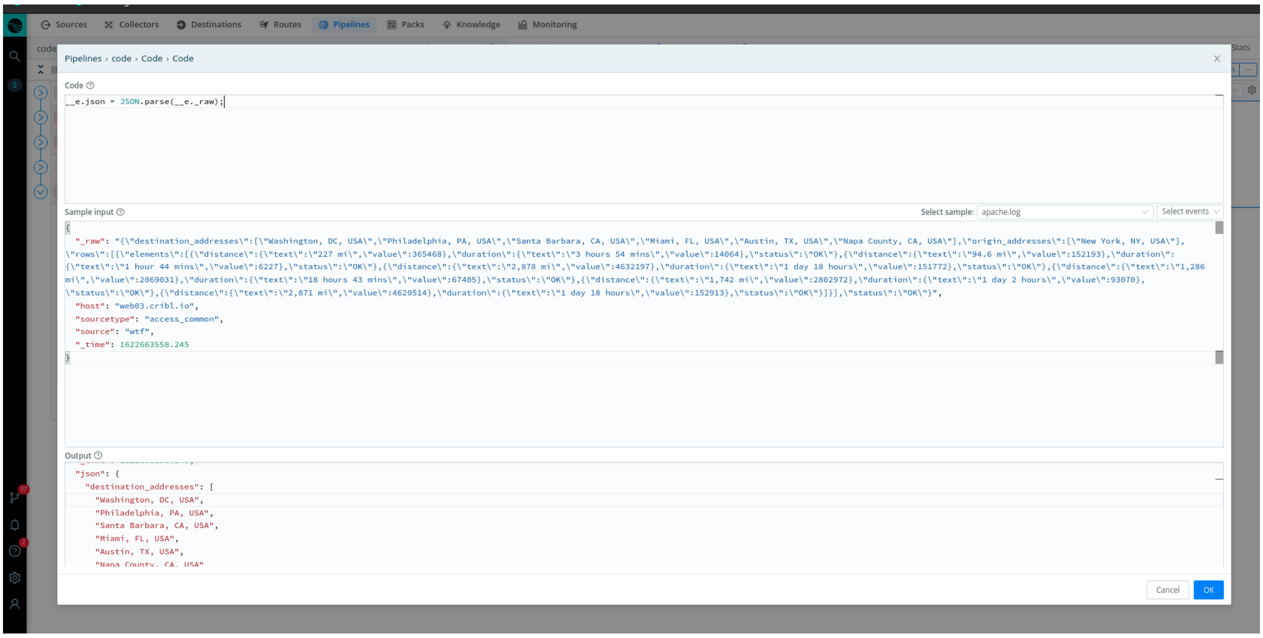


	Channel ↑	Level
□	func:code	debug

Debugging a Code Function with regular logs

Previewing

By using the expanded editor, users can run the expression against a sample event, and can preview the transformation:



Code

```
__e.json = JSON.parse(__e._raw);
```

Sample input

Select sample: apache.log

Select events

```
{
  "_raw": "{\"destination_addresses\":[\"Washington, DC, USA\",\"Philadelphia, PA, USA\",\"Santa Barbara, CA, USA\",\"Miami, FL, USA\",\"Austin, TX, USA\",\"Napa County, CA, USA\"],\"origin_addresses\":[\"New York, NY, USA\"],
\"rows\":[{\"elements\":[{\"distance\":{\"text\":\"227 mi\",\"value\":365468},\"duration\":{\"text\":\"3 hours 54 mins\",\"value\":14064},\"status\":\"OK\"},{\"distance\":{\"text\":\"94.6 mi\",\"value\":152193},\"duration\":
{\"text\":\"1 hour 44 mins\",\"value\":6227},\"status\":\"OK\"},{\"distance\":{\"text\":\"2,878 mi\",\"value\":4632197},\"duration\":{\"text\":\"1 day 18 hours\",\"value\":151772},\"status\":\"OK\"},{\"distance\":{\"text\":\"1,286
mi\",\"value\":2069031},\"duration\":{\"text\":\"18 hours 43 mins\",\"value\":67405},\"status\":\"OK\"},{\"distance\":{\"text\":\"1,742 mi\",\"value\":2802972},\"duration\":{\"text\":\"1 day 2 hours\",\"value\":93070},
\"status\":\"OK\"},{\"distance\":{\"text\":\"2,871 mi\",\"value\":4620514},\"duration\":{\"text\":\"1 day 18 hours\",\"value\":152913},\"status\":\"OK\"}]}],\"status\":\"OK\"}",
  "host": "web03.cribl.io",
  "sourcetype": "access_common",
  "source": "wtf",
  "_time": 1622663558.245
}
```

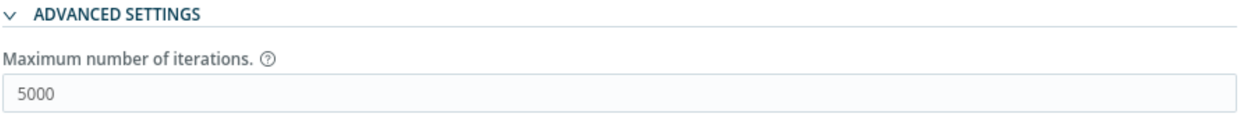
Output

```
  "json": {
    "destination_addresses": [
      "Washington, DC, USA",
      "Philadelphia, PA, USA",
      "Santa Barbara, CA, USA",
      "Miami, FL, USA",
      "Austin, TX, USA",
      "Napa County, CA, USA"
```

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.



Maximum number of iterations.: 5000

Infinite-Loop protection in a Code Function

Loops

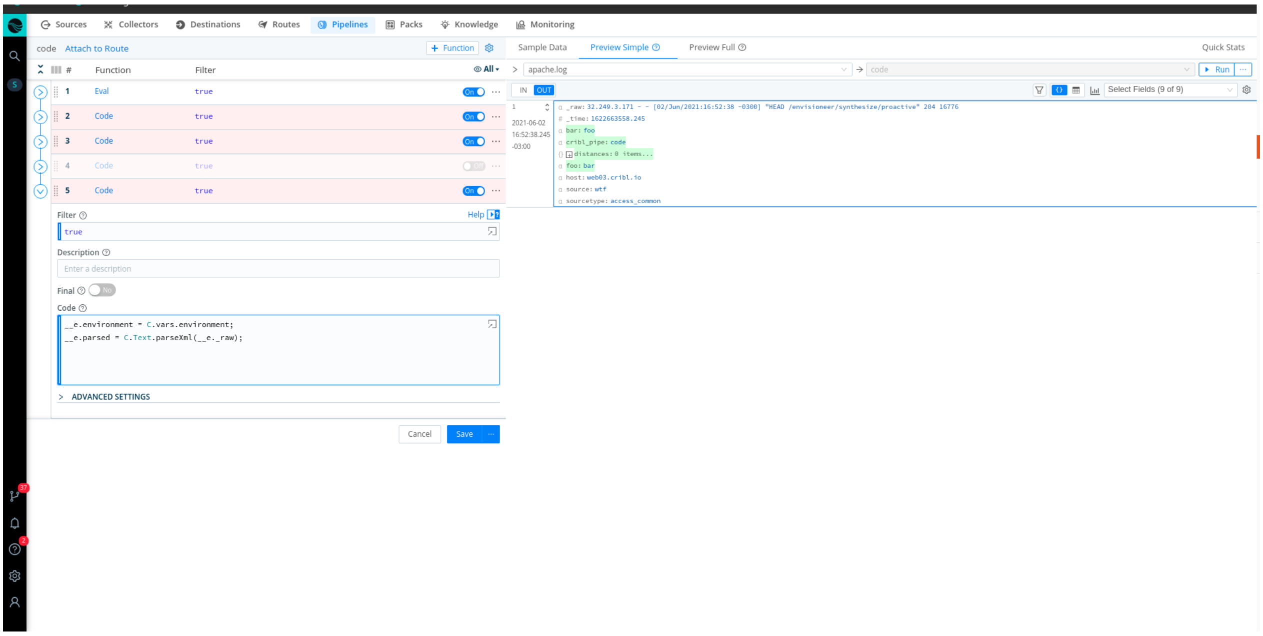
All JavaScript loops and statements are allowed: `for`, `for-of`, `while`, `do-while`, `switch`, etc.

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.



#	Function	Filter
1	Eval	true
2	Code	true
3	Code	true
4	Code	true
5	Code	true

Filter: `true`

Description: Enter a description

Final: No

Code:

```
__e.environment = C.vars.environment;
__e.parsed = C.Text.parseXml(__e._raw);
```

> ADVANCED SETTINGS

Cancel | Save

```
1
2021-06-02
16:52:38.245
-03:00
_raw: 32.249.3.171 - - [02/Jun/2021:16:52:38 -0300] "HEAD /envisioneer/synthesize/proactive" 204 16776
_time: 1622663558.245
bar: foo
cribl_pipe: code
distances: 0 items...
foo: bar
host: web03.cribl.io
source: wtf
sourcetype: access_common
```

The global `C.*` object of a Code Function

;

11.5.2. Ingest-time Fields

Adding Fields to Data in Motion

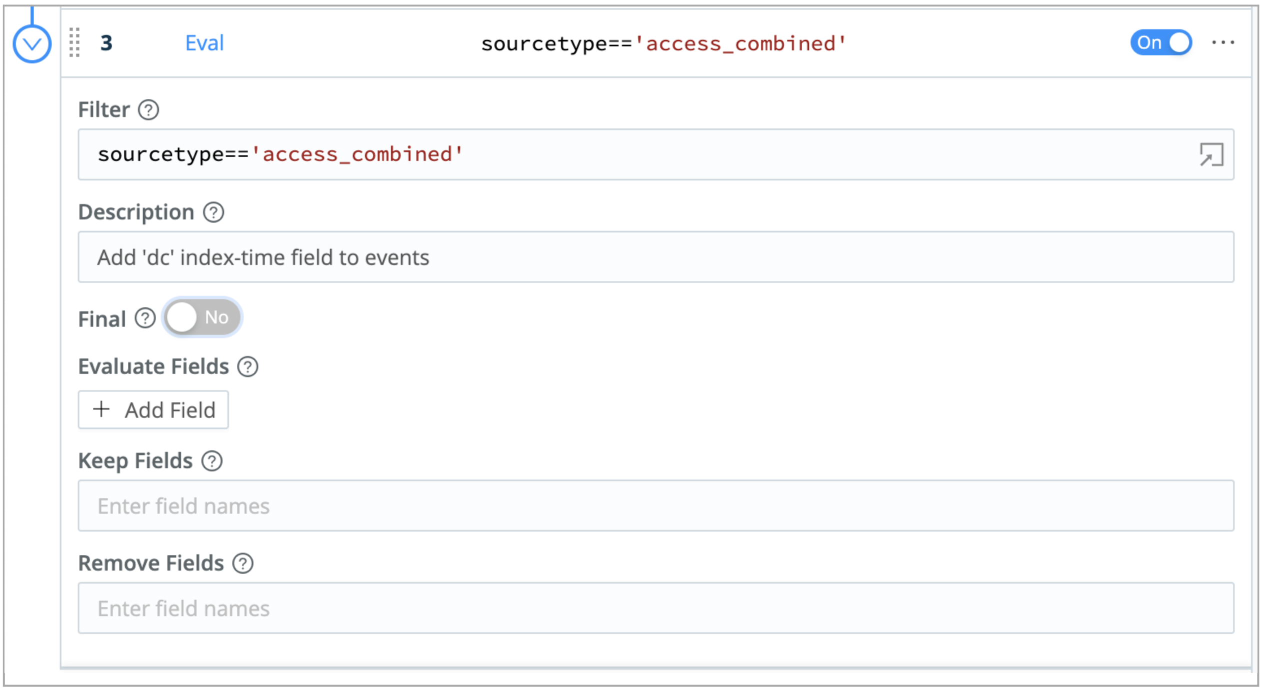
To add new fields to any event, we use the out-of-the-box **Eval** Function. We can either apply a Filter to select the events, or we can use the default `true` Filter expression to apply the Function to all incoming events.

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 a **Eval** function to it:



3 Eval `sourcetype=='access_combined'` On

Filter: `sourcetype=='access_combined'`

Description: Add 'dc' index-time field to events

Final: No

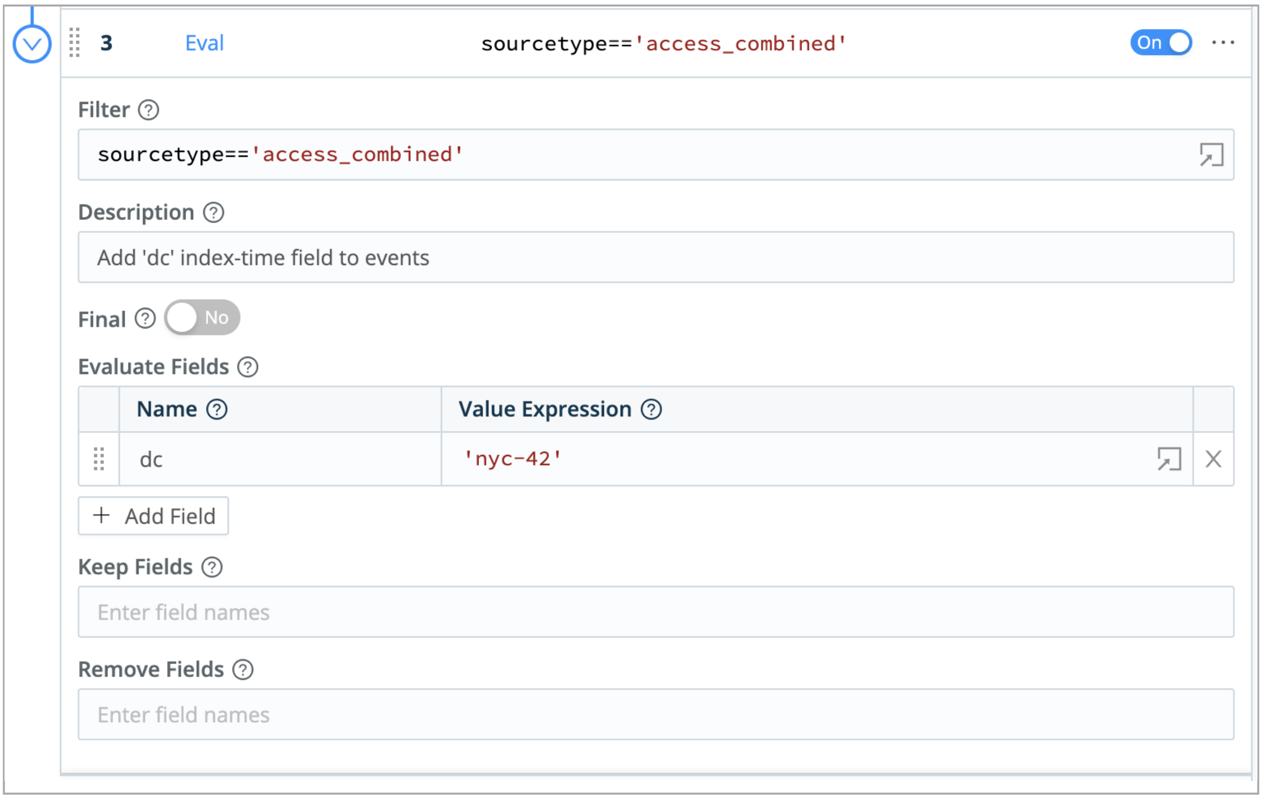
Evaluate Fields

Keep Fields: Enter field names

Remove Fields: Enter field names

Defining the Eval Function's filter expression

- Next, let's click on **+ Add Field**, add our `dc` field, and click **Save**.



sourcetype=='access_combined'

Filter: `sourcetype=='access_combined'`

Description: Add 'dc' index-time field to events

Final: No

Evaluate Fields

Name	Value Expression
dc	'nyc-42'

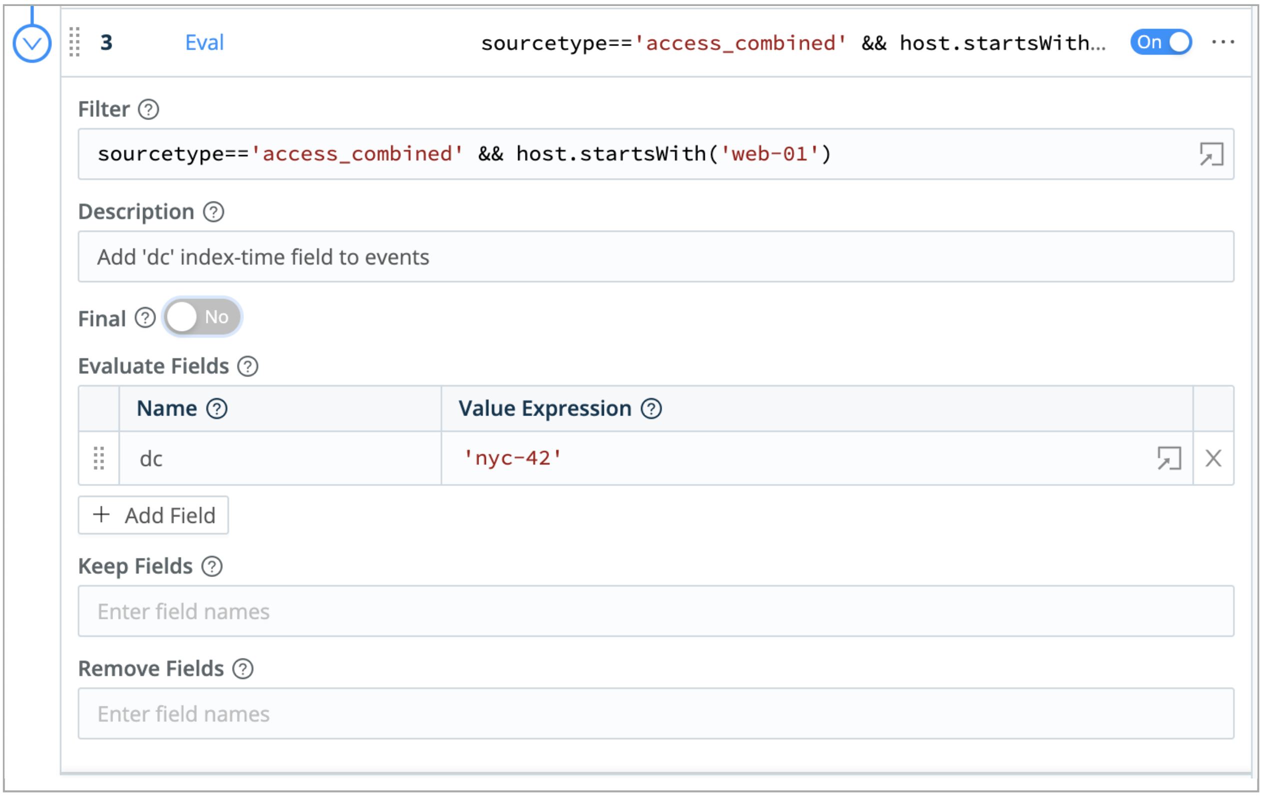
Keep Fields: Enter field names

Remove Fields: Enter field names

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 as below:



sourcetype=='access_combined' && host.startsWith...

Filter: `sourcetype=='access_combined' && host.startsWith('web-01')`

Description: Add 'dc' index-time field to events

Final: No

Evaluate Fields

Name	Value Expression
dc	'nyc-42'

Keep Fields: Enter field names

Remove Fields: Enter field names

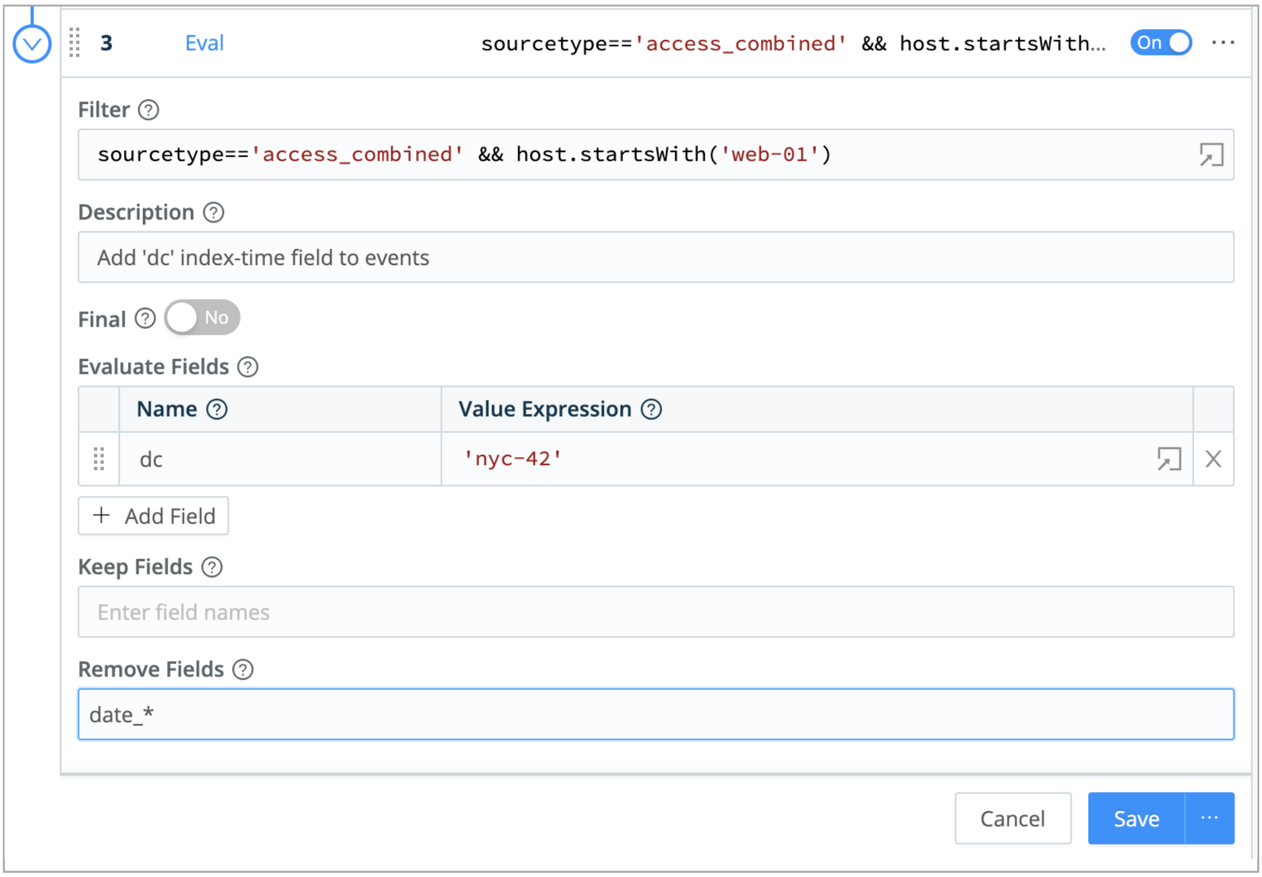
Refining the filter

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 a **Eval** function to it (as above).
- Next, in **Remove Fields**, add `date_*` and hit Save.



3 Eval `sourcetype=='access_combined' && host.startsWith...` On

Filter

```
sourcetype=='access_combined' && host.startsWith('web-01')
```

Description

Add 'dc' index-time field to events

Final: No

Evaluate Fields

Name	Value Expression
dc	'nyc-42'

Keep Fields

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!

;

11.5.3. 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.

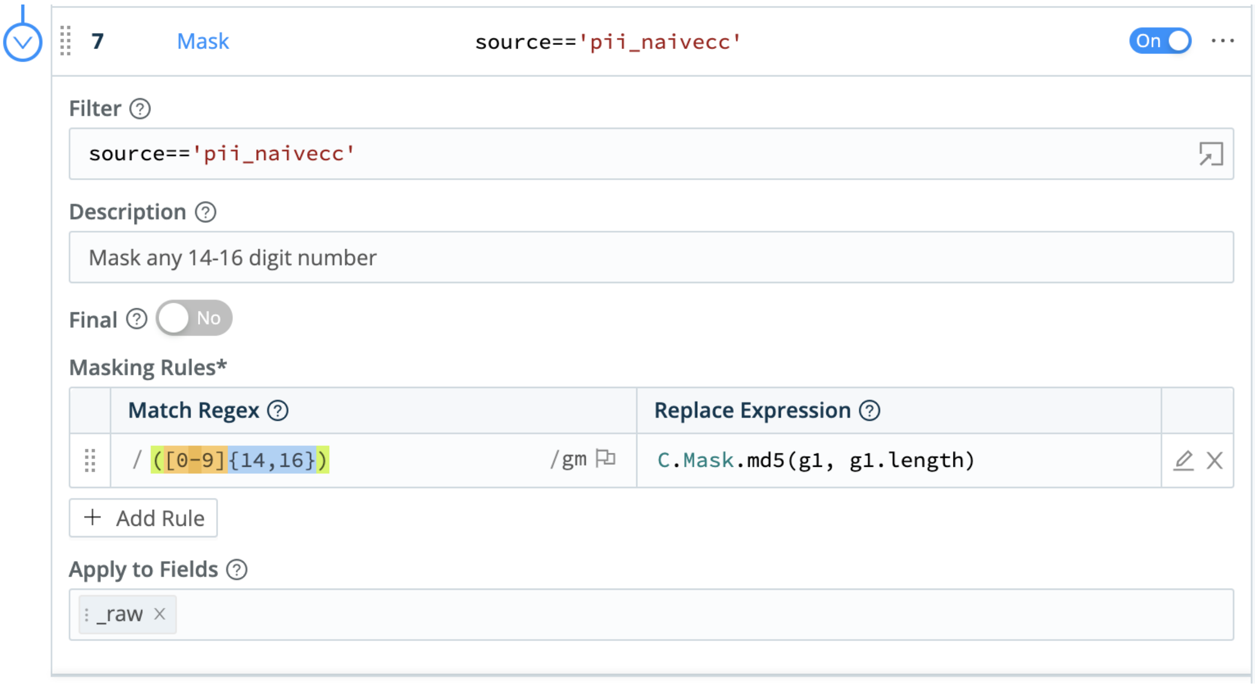
Replace Expression is a JS expression or literal to replace matched content.

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:



source=='pii_naivecc'

Filter: source=='pii_naivecc'

Description: Mask any 14-16 digit number

Final: No

Masking Rules*

Match Regex	Replace Expression
/([0-9]{14,16})/gm	C.Mask.md5(g1, g1.length)

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.

;

11.5.4. 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

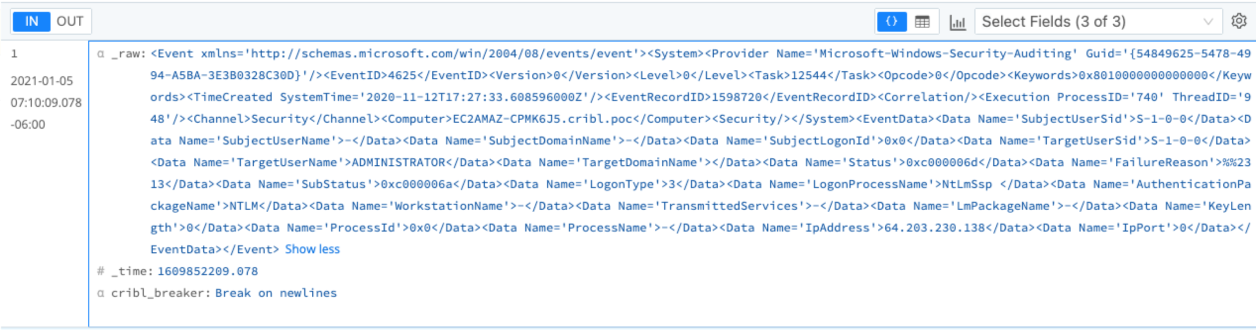
Cribl Stream's internal C.Text.parseWinEvent method parses Windows XML strings and returns a prettified JSON object. You can use this function within an Eval Function to parse an event or an ad hoc XML string. It works like C.Text.parseXml, but with Windows events, it produces more-compact output.

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.

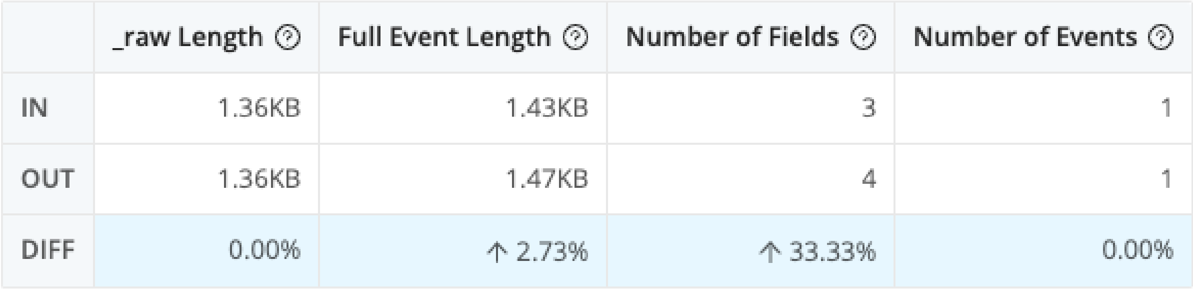
XML: Threat or Menace?

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            α _raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Security-Auditing' Guid='{54849625-5478-49
2021-01-05           94-A5BA-3E3B0328C30D}'/><EventID>4625</EventID><Version>0</Version><Level>0</Level><Task>12544</Task><Opcode>0</Opcode><Keywords>0x8010000000000000</Keyw
07:10:09.078         ords><TimeCreated SystemTime='2020-11-12T17:27:33.608596000Z'/><EventRecordID>1598720</EventRecordID><Correlation/><Execution ProcessID='740' ThreadID='9
-06:00               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
```

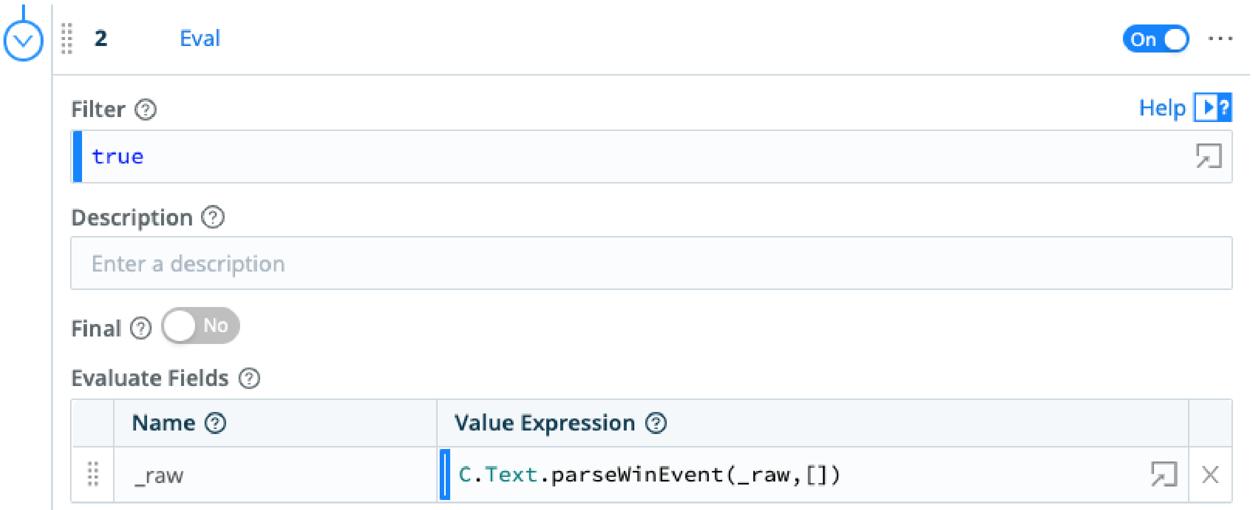
This Windows XML `_raw` event is 1.36KB in size:



	_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:



Filter ⓘ

`true`

Description ⓘ

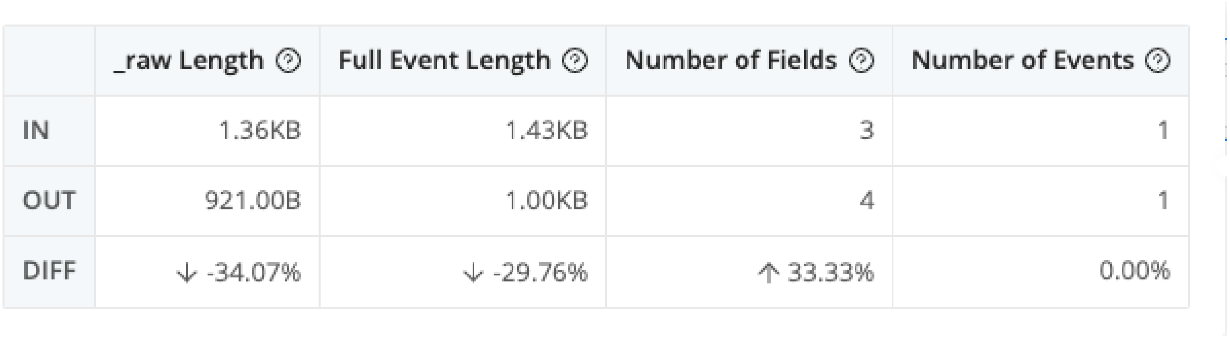
Enter a description

Final ⓘ No

Evaluate Fields ⓘ

Name ⓘ	Value Expression ⓘ
_raw	`C.Text.parseWinEvent(_raw,[])`

The resulting JSON event is now down to 921.00B in size, a 34.07% reduction of the event:

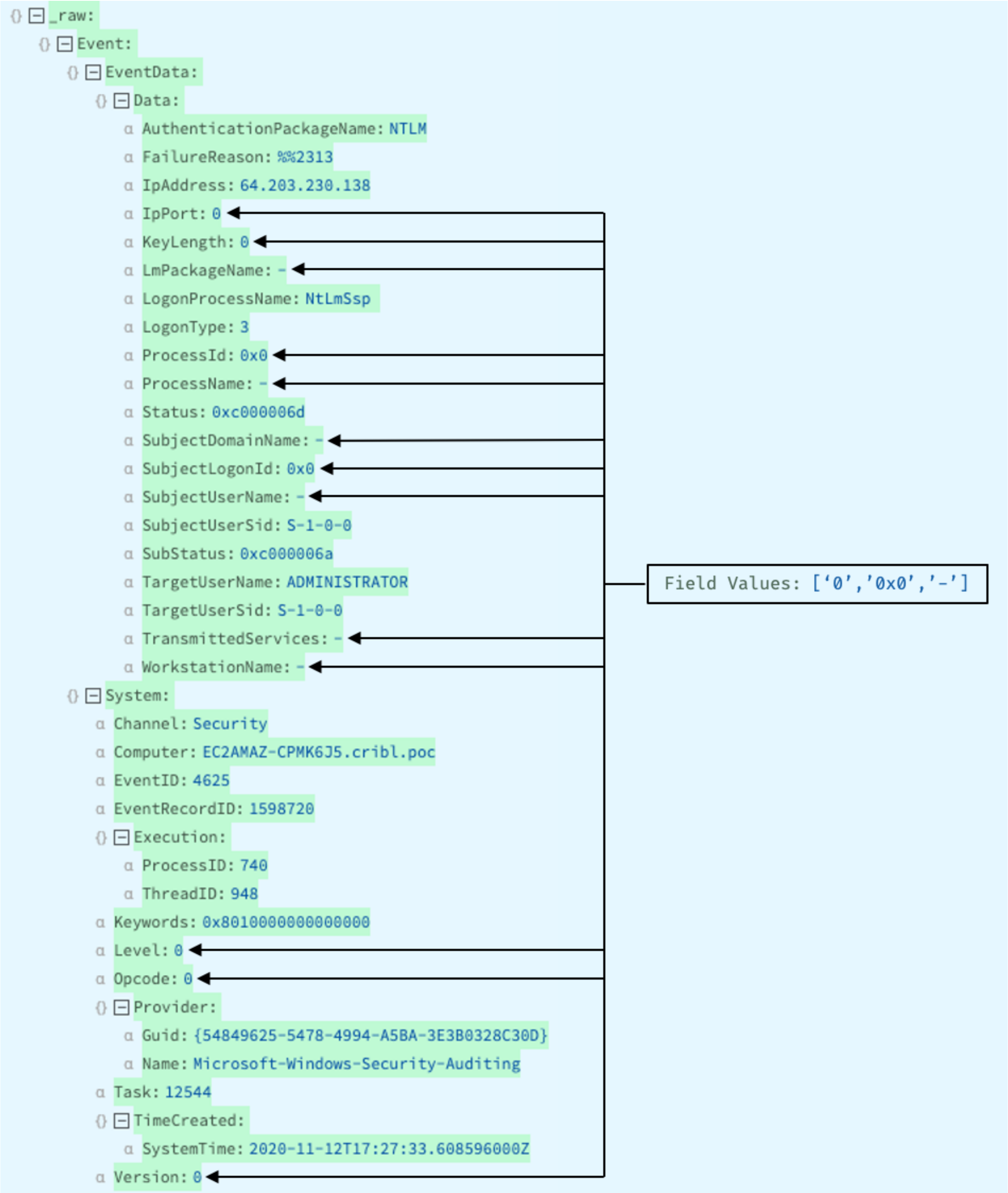


	_raw Length	Full Event Length	Number of Fields	Number of Events
IN	1.36KB	1.43KB	3	1
OUT	921.00B	1.00KB	4	1
DIFF	↓ -34.07%	↓ -29.76%	↑ 33.33%	0.00%

> 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

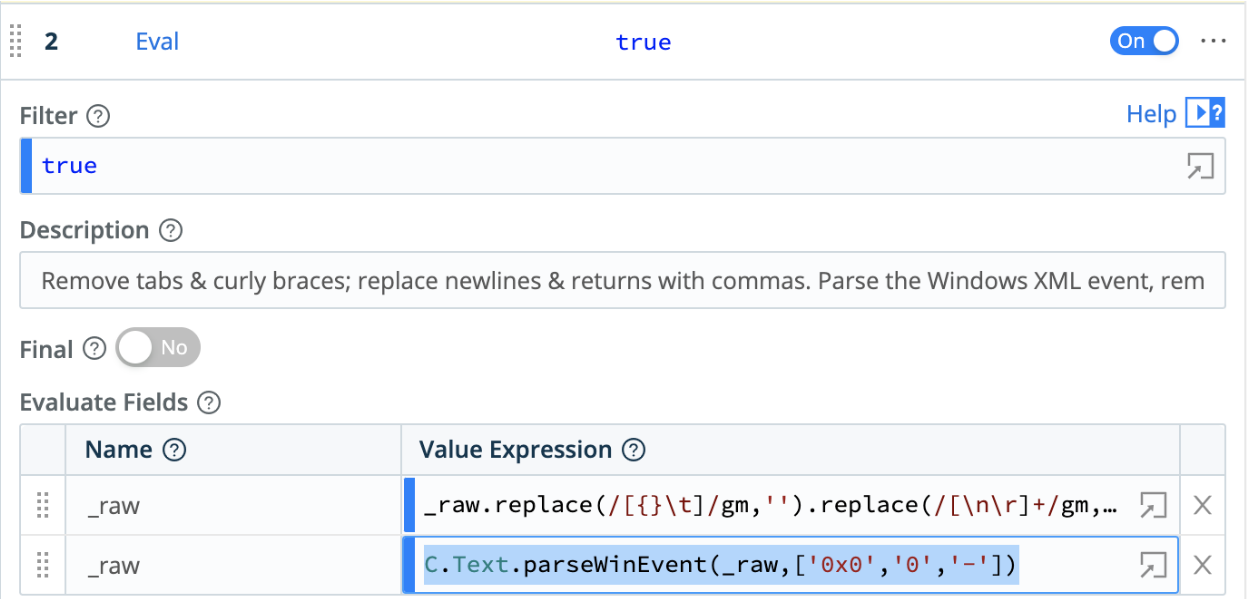
But we can do better. The fields containing essentially null values (`'0'`,`'0x0'`, or `'-'`) bloat events, demanding extra infrastructure and storage:



```
_raw:
  Event:
    EventData:
      Data:
        AuthenticationPackageName: NTLM
        FailureReason: %%2313
        IpAddress: 64.203.230.138
        IpPort: 0
        KeyLength: 0
        LmPackageName: -
        LogonProcessName: NtLmSsp
        LogonType: 3
        ProcessId: 0x0
        ProcessName: -
        Status: 0xc000006d
        SubjectDomainName: -
        SubjectLogonId: 0x0
        SubjectUserName: -
        SubjectUserSid: S-1-0-0
        SubStatus: 0xc000006a
        TargetUserName: ADMINISTRATOR
        TargetUserSid: S-1-0-0
        TransmittedServices: -
        WorkstationName: -
    System:
      Channel: Security
      Computer: EC2AMAZ-CPMK6J5.cribl.poc
      EventID: 4625
      EventRecordID: 1598720
      Execution:
        ProcessID: 740
        ThreadID: 948
      Keywords: 0x8010000000000000
      Level: 0
      Opcode: 0
      Provider:
        Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
        Name: Microsoft-Windows-Security-Auditing
      Task: 12544
      TimeCreated:
        SystemTime: 2020-11-12T17:27:33.608596000Z
      Version: 0
```

Field Values: [‘0’,’0x0’,’-’]

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:



Filter: `true`

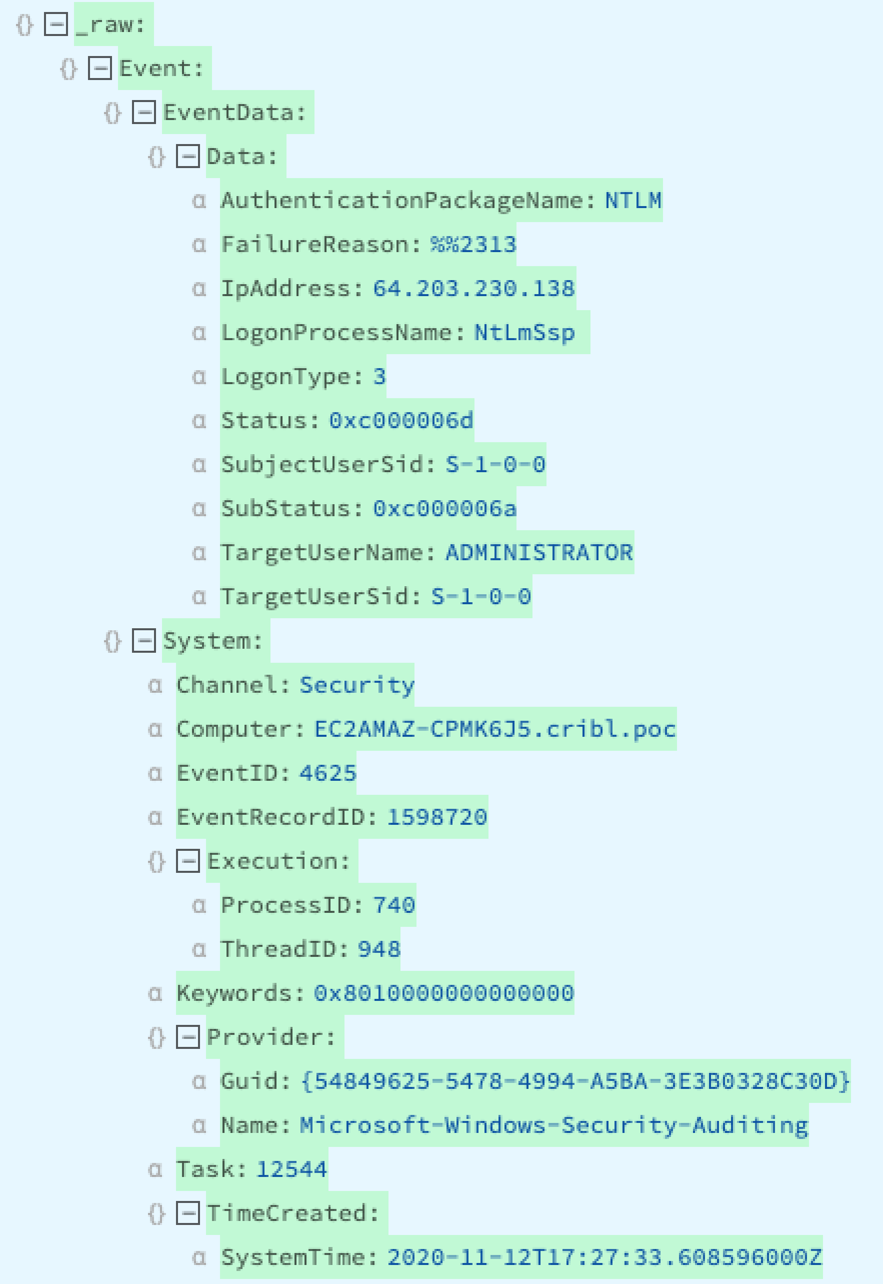
Description: Remove tabs & curly braces; replace newlines & returns with commas. Parse the Windows XML event, rem

Final: No

Evaluate Fields

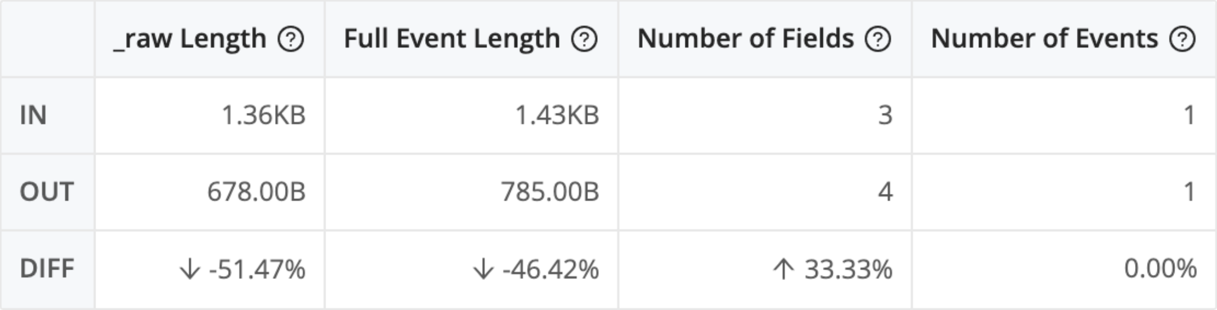
Name	Value Expression
_raw	`_raw.replace(/[{}\t]/gm,'').replace(/[\n\r]+/gm,…`
_raw	`C.Text.parseWinEvent(_raw,['0x0','0','-'])`

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:
        AuthenticationPackageName: NTLM
        FailureReason: %%2313
        IpAddress: 64.203.230.138
        LogonProcessName: NtLmSsp
        LogonType: 3
        Status: 0xc000006d
        SubjectUserSid: S-1-0-0
        SubStatus: 0xc000006a
        TargetUserName: ADMINISTRATOR
        TargetUserSid: S-1-0-0
    System:
      Channel: Security
      Computer: EC2AMAZ-CPMK6J5.cribl.poc
      EventID: 4625
      EventRecordID: 1598720
      Execution:
        ProcessID: 740
        ThreadID: 948
      Keywords: 0x8010000000000000
      Provider:
        Guid: {54849625-5478-4994-A5BA-3E3B0328C30D}
        Name: Microsoft-Windows-Security-Auditing
      Task: 12544
      TimeCreated:
        SystemTime: 2020-11-12T17:27:33.608596000Z
```

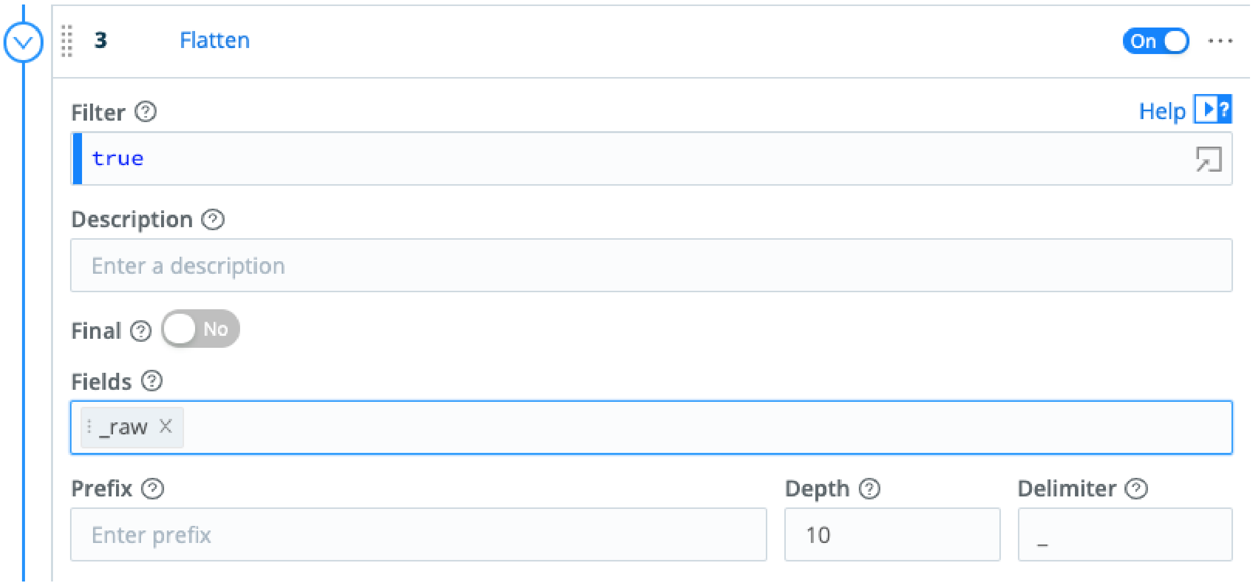
The event is now down to 678.00B in size, translating to a 51.47% reduction from the original event:



	_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:



Filter: true

Description: Enter a description

Final: No

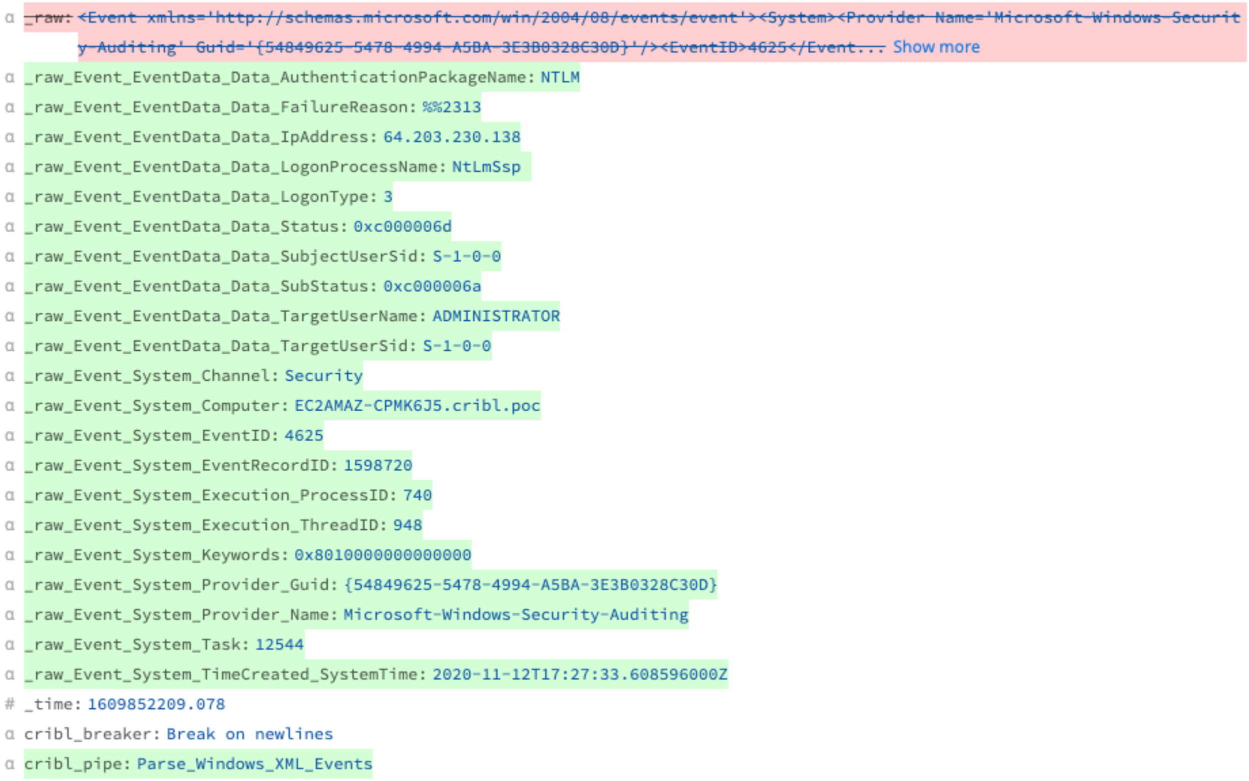
Fields: _raw

Prefix: Enter prefix

Depth: 10

Delimiter: _

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
```

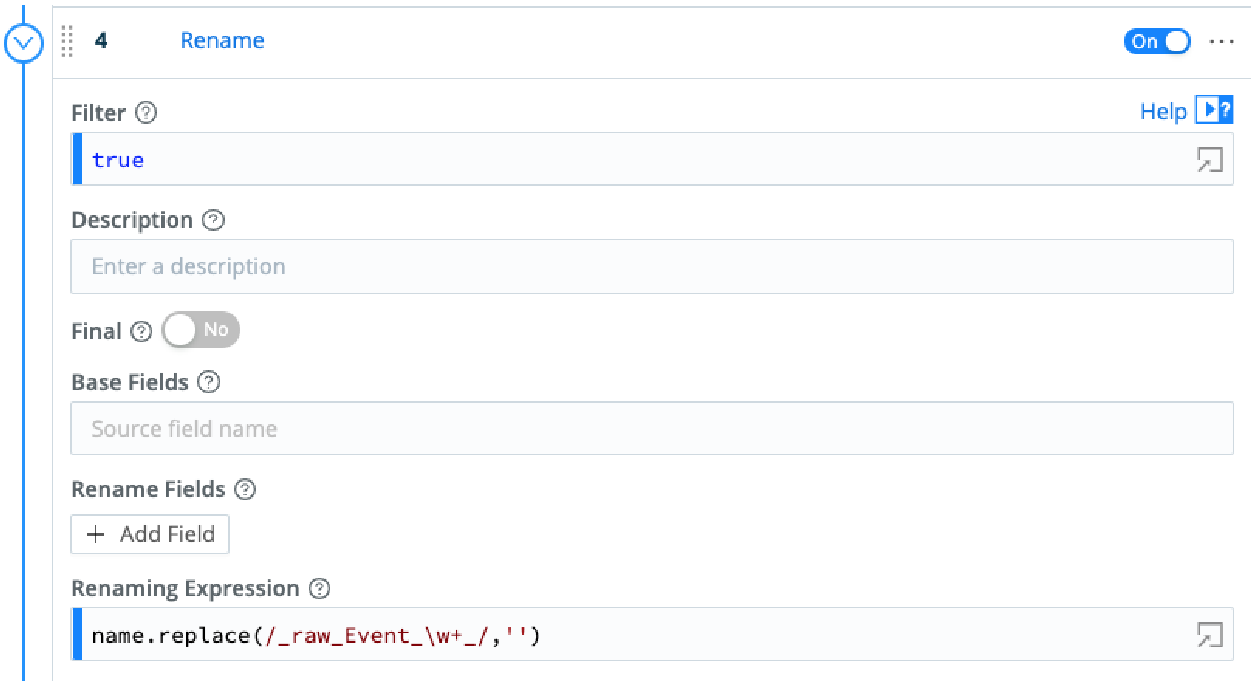
> Don't worry about the `_raw` field's deletion (red strikeout). This is the Flatten Function's default behavior. We'll restore `_raw` after we clean and reduce the event even more.

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 **Renaming Expression**, we build a JavaScript expression to match the field names' prefixes (up to the underscore):



4 Rename On

Filter: `true`

Description: Enter a description

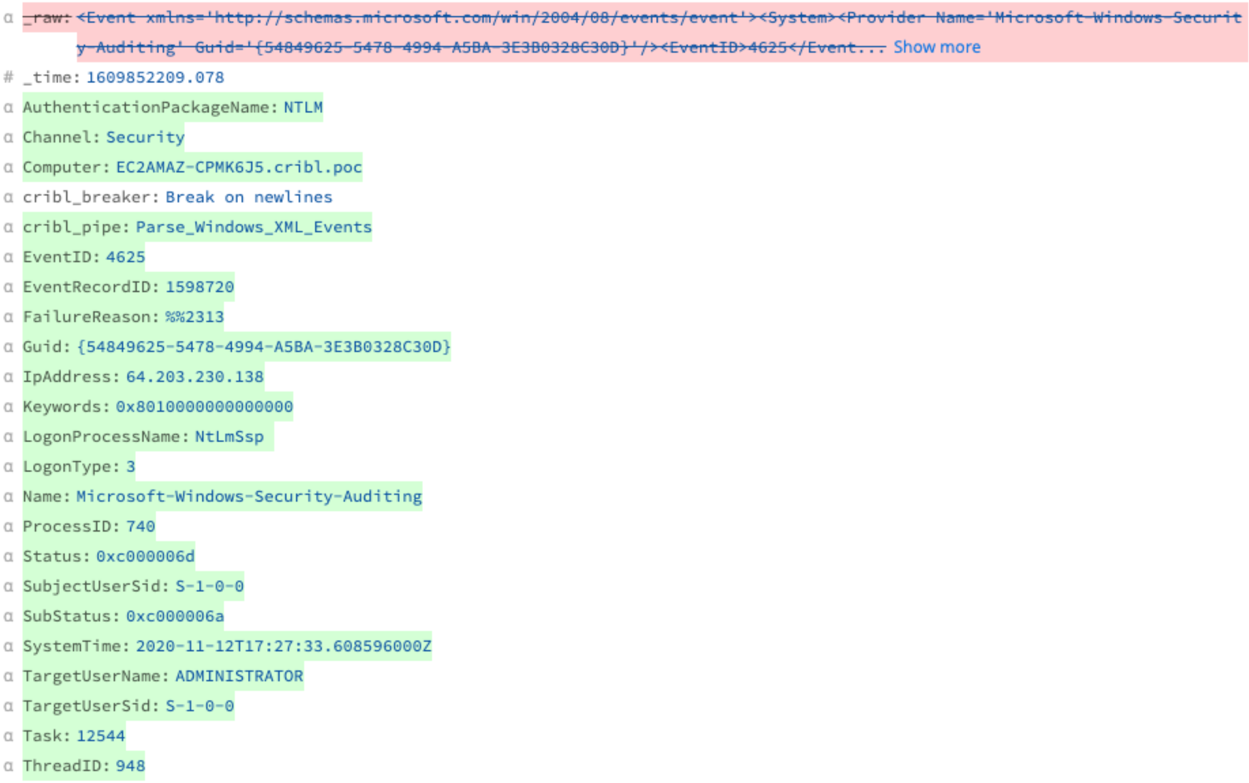
Final: No

Base Fields: Source field name

Rename Fields: + Add Field

Renaming Expression: `name.replace(/_raw_Event_\w+_/,'')`

The resulting field names are now much more compact, and easier to work with and manage:

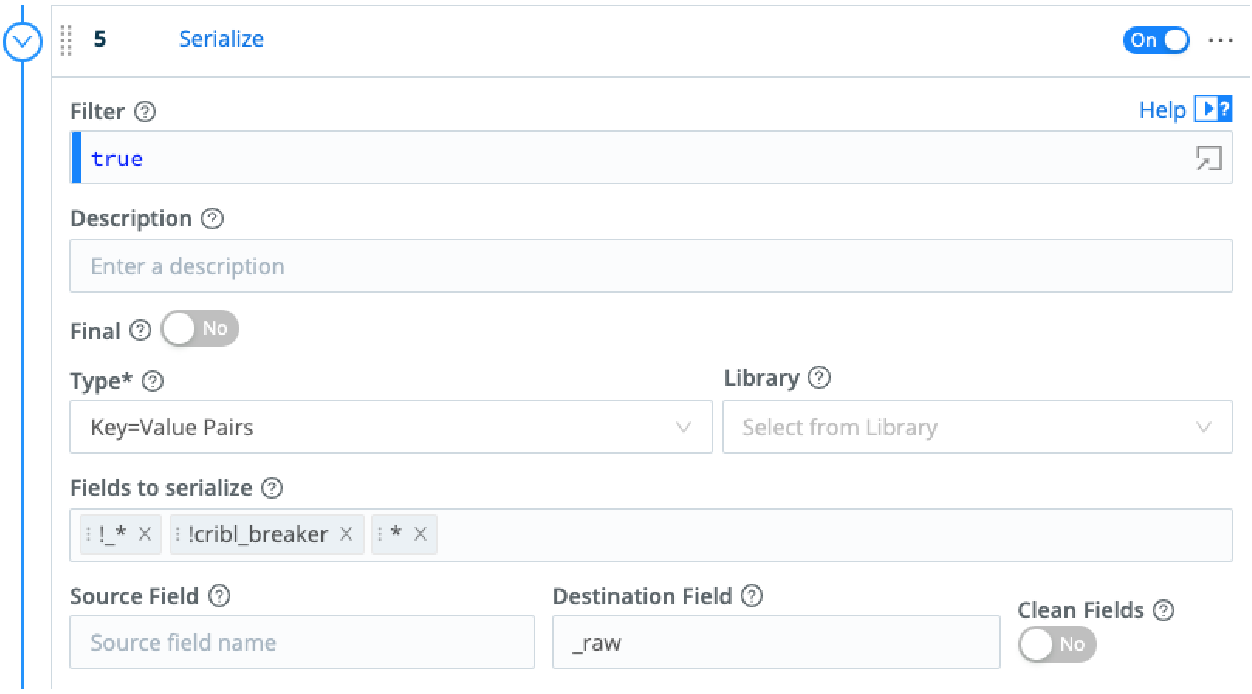


```
α ~~_raw: <Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider Name='Microsoft-Windows-Securit
      y-Auditing' Guid='{54849625-5478-4994-A5BA-3E3B0328C30D}'/><EventID>4625</Event...~~ Show more
# _time: 1609852209.078
α AuthenticationPackageName: NTLM
α Channel: 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.) Here, Serialize takes the extracted fields and puts them back into `_raw`:



5 Serialize On

Filter: true

Description: Enter a description

Final: No

Type*: Key=Value Pairs

Library: Select from Library

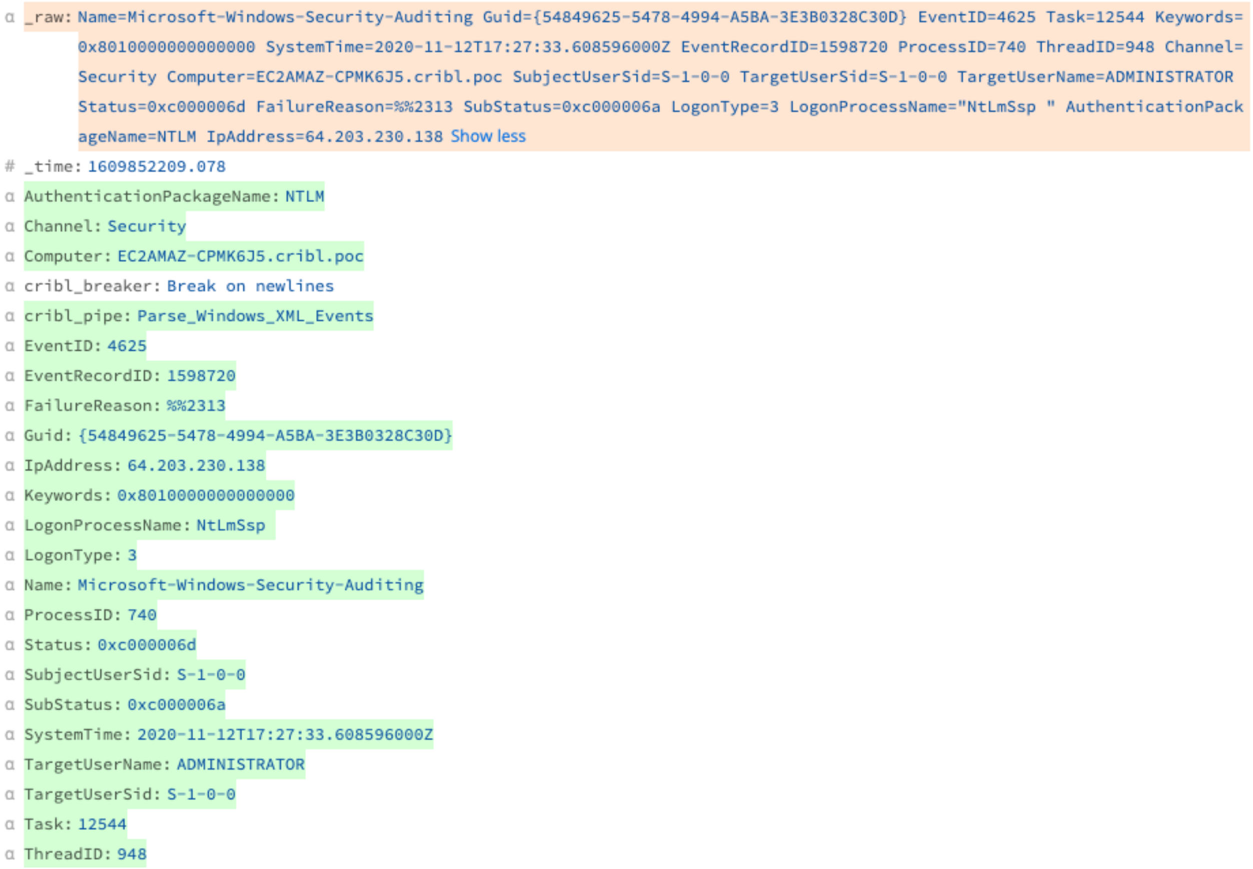
Fields to serialize: !_* !cribl_breaker *

Source Field: Source field name

Destination Field: _raw

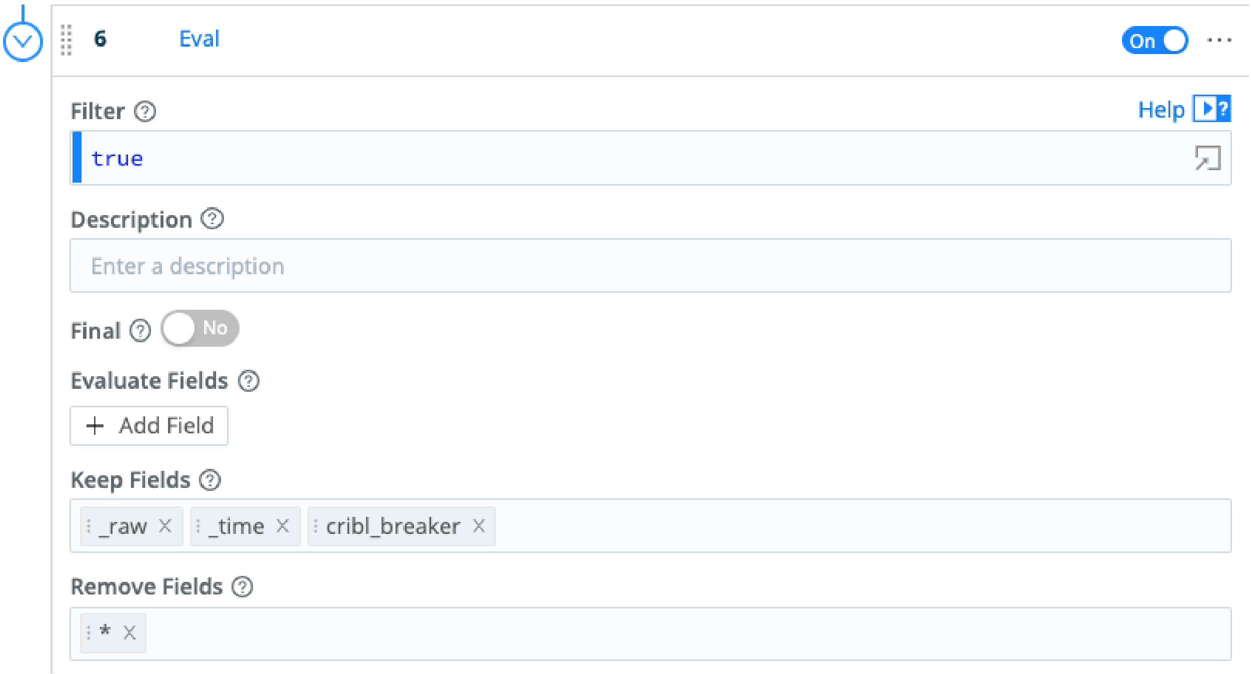
Clean Fields: No

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:



6 Eval On

Filter: true

Description: Enter a description

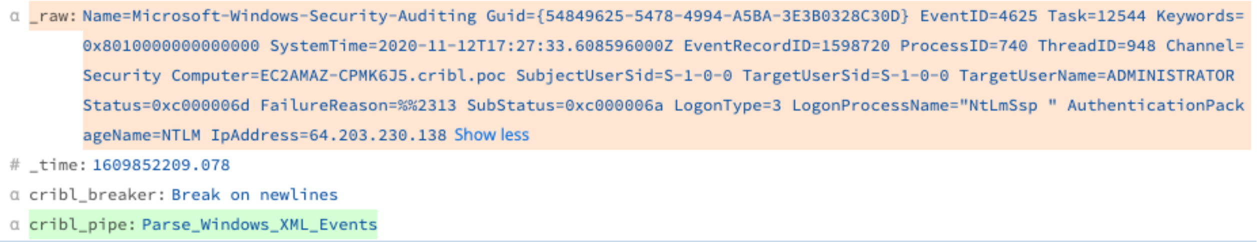
Final: No

Evaluate Fields: + Add Field

Keep Fields: _raw, _time, cribl_breaker

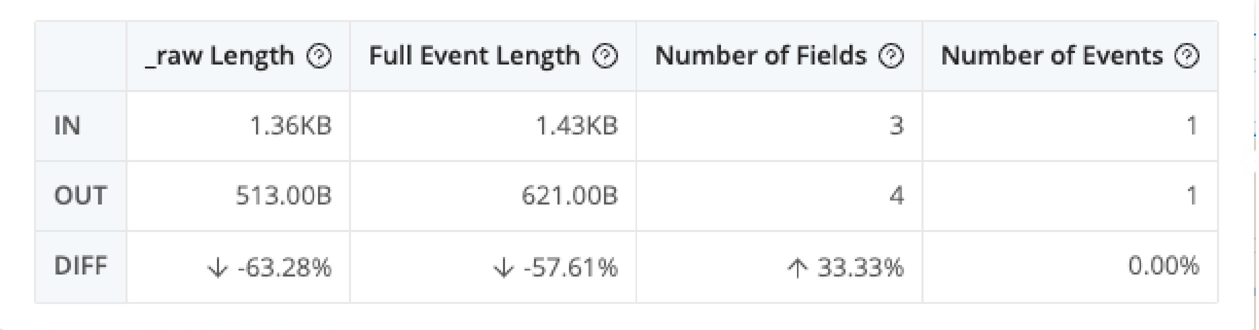
Remove Fields: *

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 events, retains full fidelity of the data, and dramatically reduces license and storage costs by up to 70%.\n\nAuthor: David Maislin - (david@cribl.io) - Find me on our Community Slack"
        },
        "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 with commas. Parse the Windows XML event, removing null or unnecessary fields & whitespace."
      },
      {
        "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",
            "*"
          ],
          "dstField": "_raw",
          "cleanFields": false
        },
        "id": "serialize",
        "disabled": false,
        "description": "Serialize top level events back into _raw in the desired
type."
      },
      {
        "filter": "true",
        "conf": {
          "keep": [
            "_time",
            "_raw"
          ],
          "remove": [
            "*"
          ]
        },
        "id": "eval",
        "disabled": false,
        "description": "Drop unnecessary fields."
      }
    ]
  }
}
```

Finally, here's a sample of Windows XML events that you can upload to Cribl Stream's Preview pane to try this out:

Sample data

```
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider
3E3B0328C30D}'/><EventID>4625</EventID><Version>0</Version><Level>0</Level><Task>12544<
SystemTime='2020-11-12T17:27:33.608596000Z'/><EventRecordID>1598720</EventRecordID><Cor
<Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></S
Name='SubjectUserName'>-</Data><Data Name='SubjectDomainName'>-</Data><Data Name='Subje
Name='TargetUserName'>ADMINISTRATOR</Data><Data Name='TargetDomainName'></Data><Data Na
Name='SubStatus'>0xc000006a</Data><Data Name='LogonType'>3</Data><Data Name='LogonProce
<Data Name='WorkstationName'>-</Data><Data Name='TransmittedServices'>-</Data><Data Nam
Name='ProcessId'>0x0</Data><Data Name='ProcessName'>-</Data><Data Name='IpAddress'>64.2
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider
3E3B0328C30D}'/><EventID>4672</EventID><Version>0</Version><Level>0</Level><Task>12548<
SystemTime='2020-11-12T17:27:33.622556400Z'/><EventRecordID>1598721</EventRecordID><Cor
<Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></S
Name='SubjectUserName'>EC2AMAZ-CPMK6J5$</Data><Data Name='SubjectDomainName'>CRIBL</Dat
Name='PrivilegeList'>SeSecurityPrivilege
            SeBackupPrivilege
            SeRestorePrivilege
            SeTakeOwnershipPrivilege
            SeDebugPrivilege
            SeSystemEnvironmentPrivilege
            SeLoadDriverPrivilege
            SeImpersonatePrivilege
            SeDelegateSessionUserImpersonatePrivilege
            SeEnableDelegationPrivilege</Data></EventData></Event>
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider
3E3B0328C30D}'/><EventID>4624</EventID><Version>2</Version><Level>0</Level><Task>12544<
SystemTime='2020-11-12T17:27:33.622676800Z'/><EventRecordID>1598722</EventRecordID><Cor
<Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></S
Name='SubjectUserName'>-</Data><Data Name='SubjectDomainName'>-</Data><Data Name='Subje
Name='TargetUserName'>EC2AMAZ-CPMK6J5$</Data><Data Name='TargetDomainName'>CRIBL.POC</D
Name='LogonType'>3</Data><Data Name='LogonProcessName'>Kerberos</Data><Data Name='Authe
<Data Name='LogonGuid'>{A9F7AB8C-F5DA-B736-AF84-F1A02DFA3FA4}</Data><Data Name='Transmi
Name='KeyLength'>0</Data><Data Name='ProcessId'>0x0</Data><Data Name='ProcessName'>-</D
Name='ImpersonationLevel'>%%1833</Data><Data Name='RestrictedAdminMode'>-</Data><Data N
Name='TargetOutboundDomainName'>-</Data><Data Name='VirtualAccount'>%%1843</Data><Data
Name='ElevatedToken'>%%1842</Data></EventData></Event>
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider
3E3B0328C30D}'/><EventID>4634</EventID><Version>0</Version><Level>0</Level><Task>12545<
SystemTime='2020-11-12T17:27:33.624089200Z'/><EventRecordID>1598723</EventRecordID><Cor
<Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></S
Name='TargetUserName'>EC2AMAZ-CPMK6J5$</Data><Data Name='TargetDomainName'>CRIBL</Data>
</EventData></Event>
<Event xmlns='http://schemas.microsoft.com/win/2004/08/events/event'><System><Provider
3E3B0328C30D}'/><EventID>4625</EventID><Version>0</Version><Level>0</Level><Task>12544<
SystemTime='2020-11-12T17:27:36.185955600Z'/><EventRecordID>1598724</EventRecordID><Cor
<Channel>Security</Channel><Computer>EC2AMAZ-CPMK6J5.cribl.poc</Computer><Security/></S
Name='SubjectUserName'>-</Data><Data Name='SubjectDomainName'>-</Data><Data Name='Subje
Name='TargetUserName'>SCAN</Data><Data Name='TargetDomainName'></Data><Data Name='Statu
Name='SubStatus'>0xc0000064</Data><Data Name='LogonType'>3</Data><Data Name='LogonProce
<Data Name='WorkstationName'>-</Data><Data Name='TransmittedServices'>-</Data><Data Nam
Name='ProcessId'>0x0</Data><Data Name='ProcessName'>-</Data><Data Name='IpAddress'>163.
```

;

11.5.5. Regex Filtering

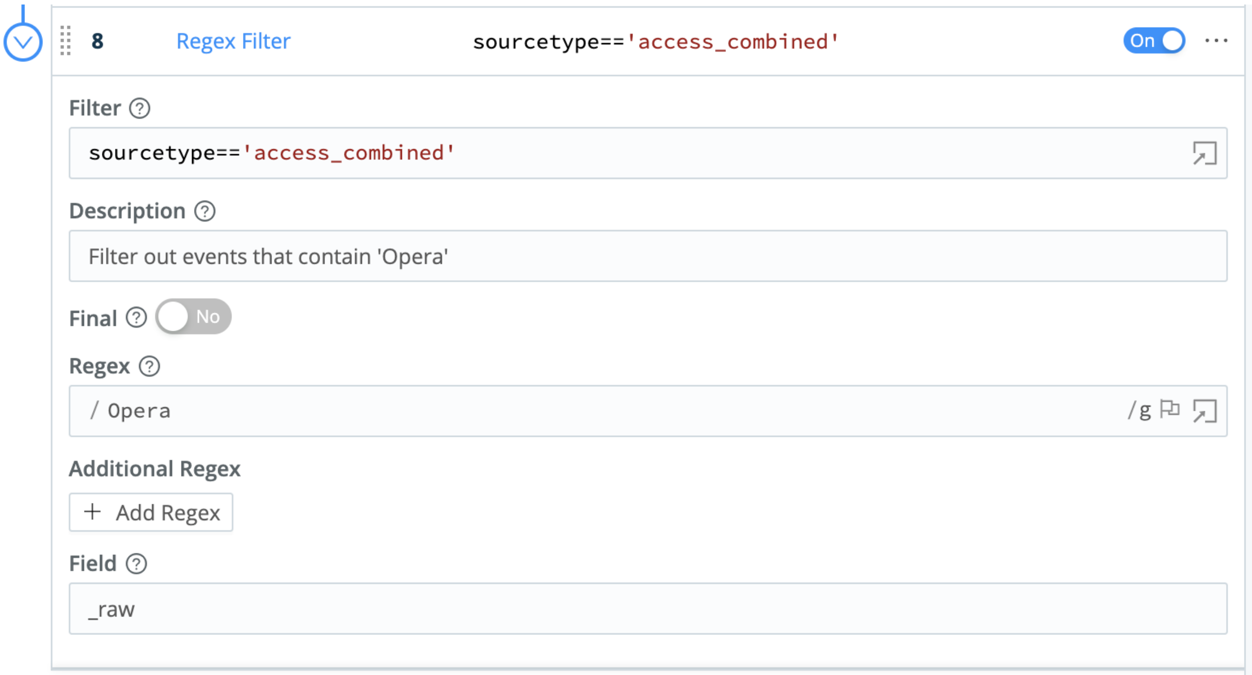
To filter events in real time (data in motion), we use the out-of-the-box **Regex Filter** Function. This is similar to nullqueueing with TRANSFORMS in Splunk, but the matching condition is way more flexible.

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:



8 Regex Filter `sourcetype=='access_combined'` On

Filter: `sourcetype=='access_combined'`

Description: Filter out events that contain 'Opera'

Final: No

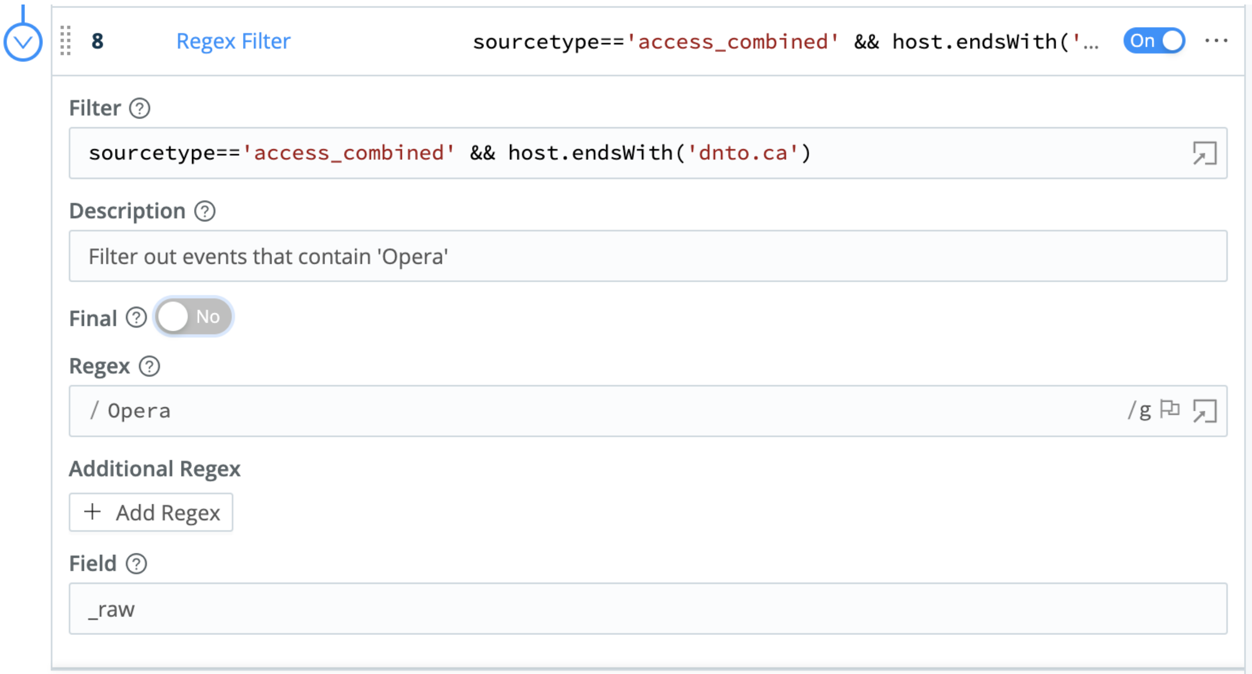
Regex: `/ Opera /g`

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 as shown below:



```
sourcetype=='access_combined' && host.endsWith('...
```

Filter: `sourcetype=='access_combined' && host.endsWith('dnto.ca')`

Description: Filter out events that contain 'Opera'

Final: No

Regex: `/ Opera /g`

Additional Regex

Field: _raw

Filtering by host

This is a very flexible method for filtering incoming events in real time, on virtually any arbitrary conditions.

;

11.6. Using Collectors

11.6.1. 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.

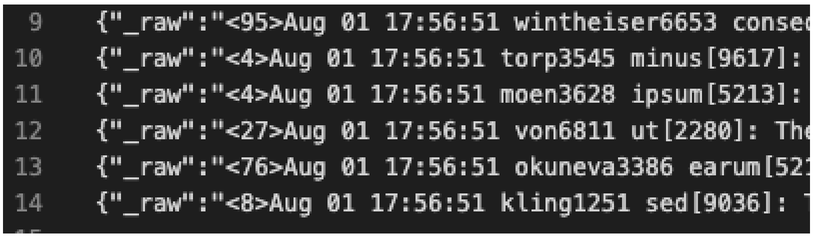
For simplicity, we'll treat the storage destination here as Amazon S3, although it could just as easily be MinIO, or any of several other options.

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:

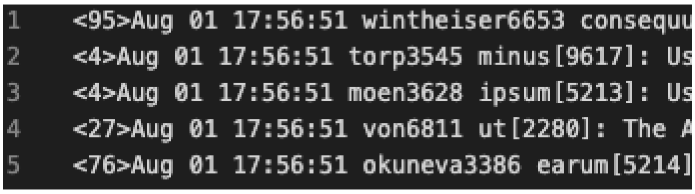


```
{"_raw":"<95>Aug 01 17:56:51 wintheiser6653 conse
{"_raw":"<4>Aug 01 17:56:51 torp3545 minus[9617]:
{"_raw":"<4>Aug 01 17:56:51 moen3628 ipsum[5213]:
{"_raw":"<27>Aug 01 17:56:51 von6811 ut[2280]: Th
{"_raw":"<76>Aug 01 17:56:51 okuneva3386 earum[52
{"_raw":"<8>Aug 01 17:56:51 kling1251 sed[9036]:
```

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 consequu
<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.

> The S3 Collector does not (as of v.3.5.3) support ingesting data in the Parquet format. Therefore, you currently cannot replay data that has been exported as Parquet.

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.

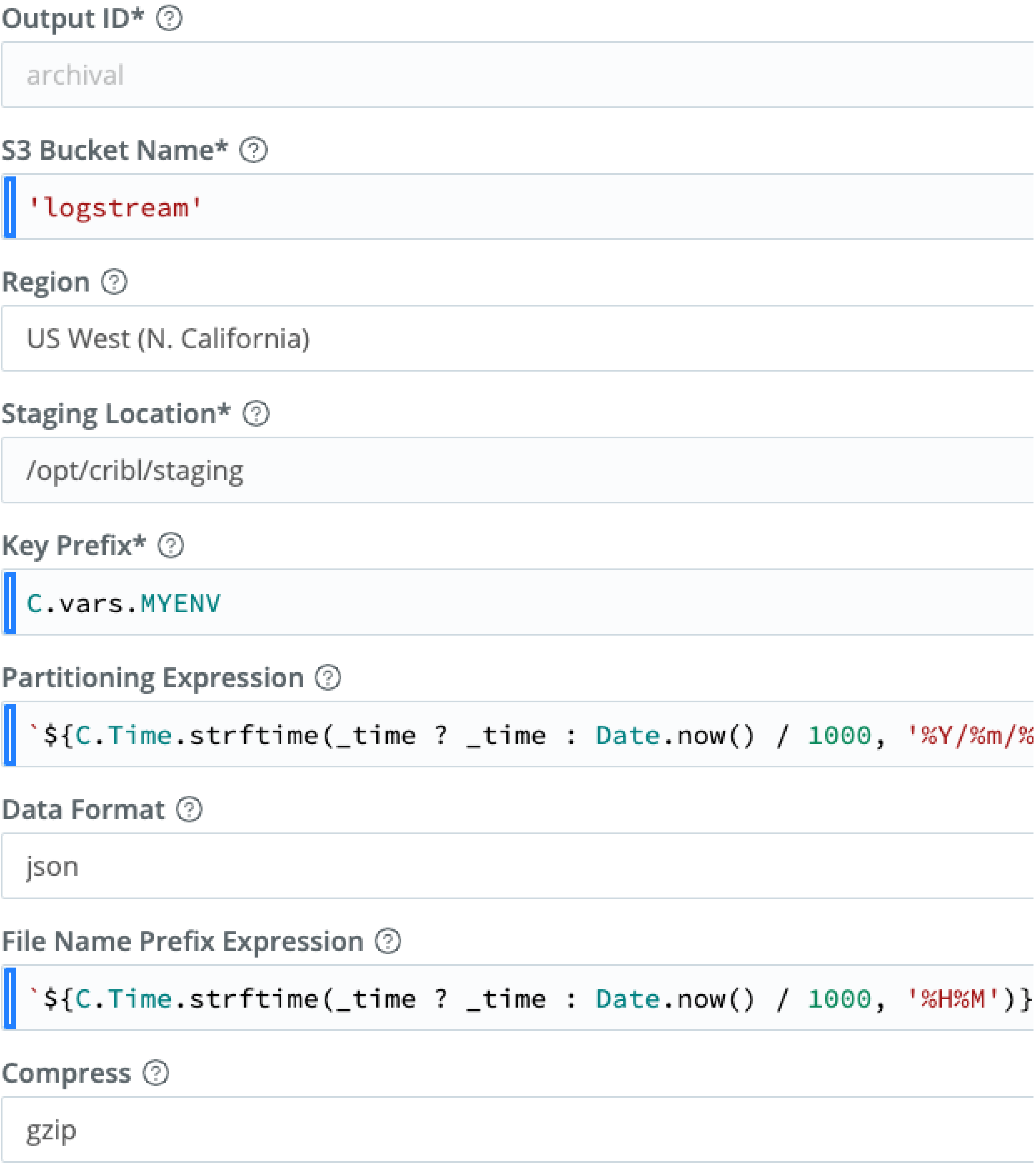
> Cribl Stream Collectors and Replay features are not compatible with "deep-freeze" storage classes that have long retrieval times. This excludes the S3 Glacier and Deep Glacier storage tiers, and also excludes Azure's archive tier.

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 : 
'no_index'}/${host ? host : 'no_host'}/${sourcetype ? sourcetype : 'no_sourcetype'}`
```

Filename:

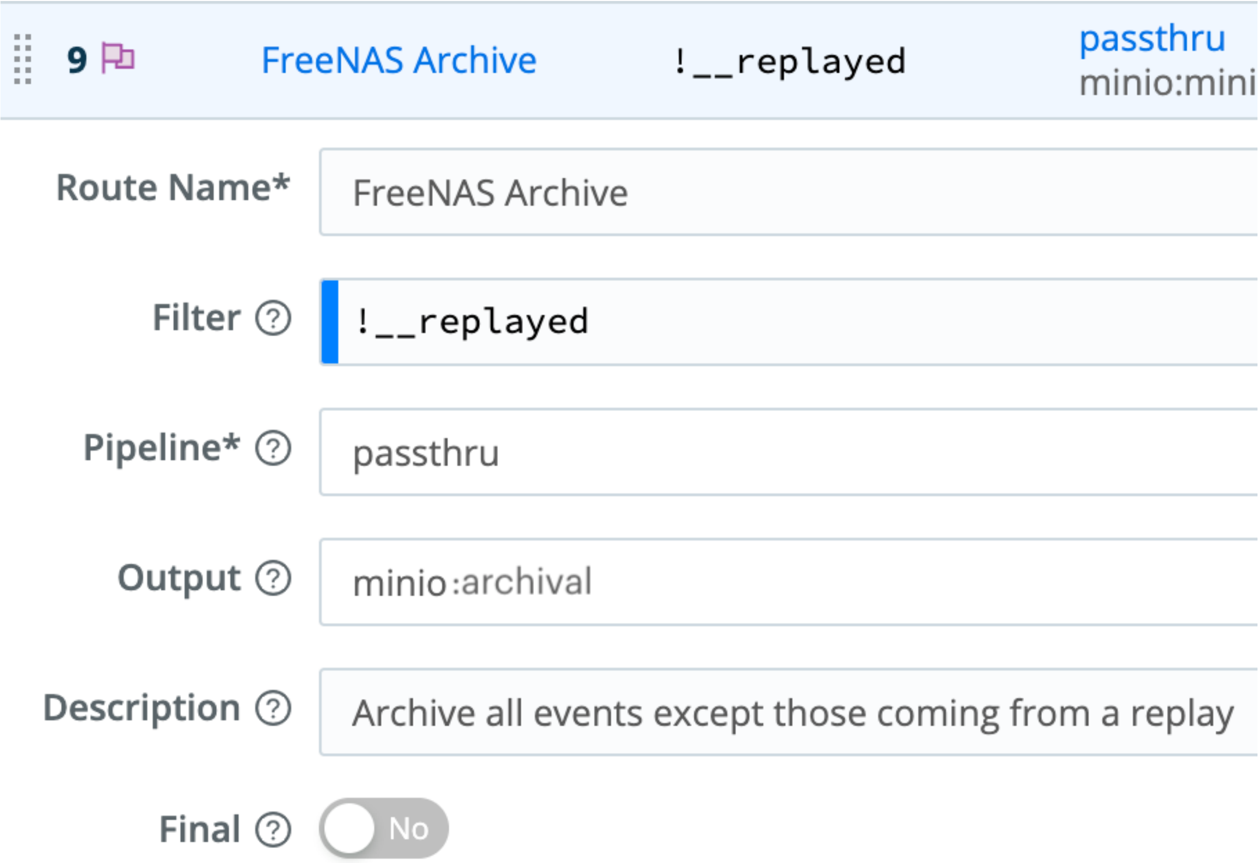
```
`${C.Time.strftime(_time ? _time : Date.now() / 1000, '%H%M')}`
```

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. But this is optional, not a required field.

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.



9 FreeNAS Archive !__replayed passthru minio:mini

Route Name* FreeNAS Archive

Filter !__replayed

Pipeline* passthru

Output minio:archival

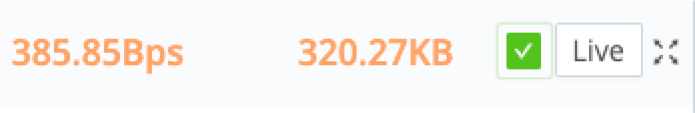
Description Archive all events except those coming from a replay

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.)



385.85Bps 320.27KB ✓ Live

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:

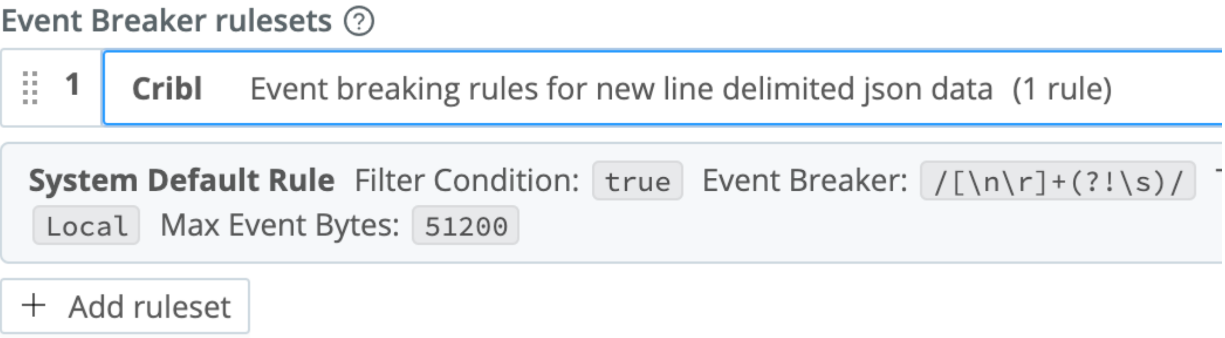
```
/${MYENV}/${_time:%Y}/${_time:%m}/${_time:%d}/${index}/${host}/${sourcetype}/${_time:%H
```

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.



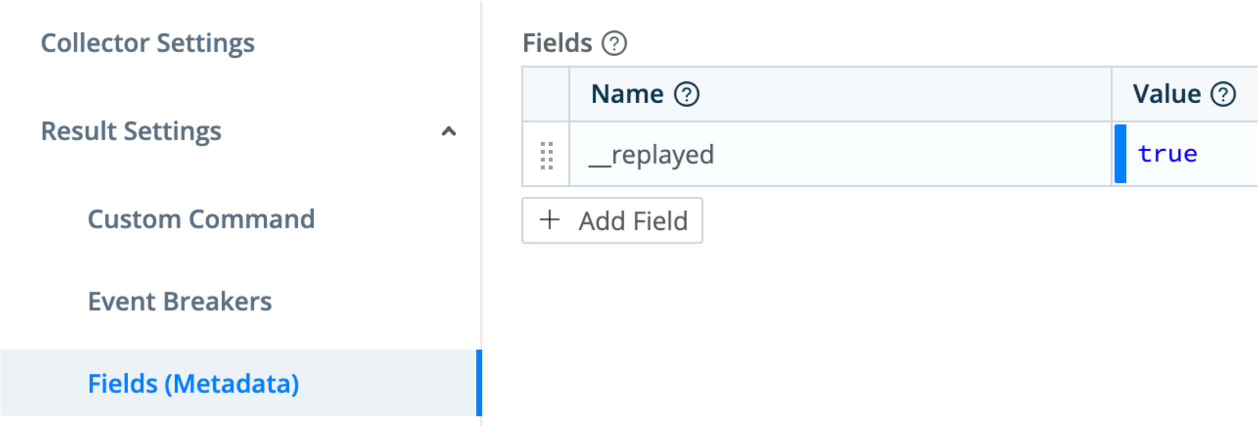
Event Breaker rulesets

1 **Cribl** Event breaking rules for new line delimited json data (1 rule)

System Default Rule Filter Condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` `Local` Max Event Bytes: `51200`

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.

;

11.6.2. 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.

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

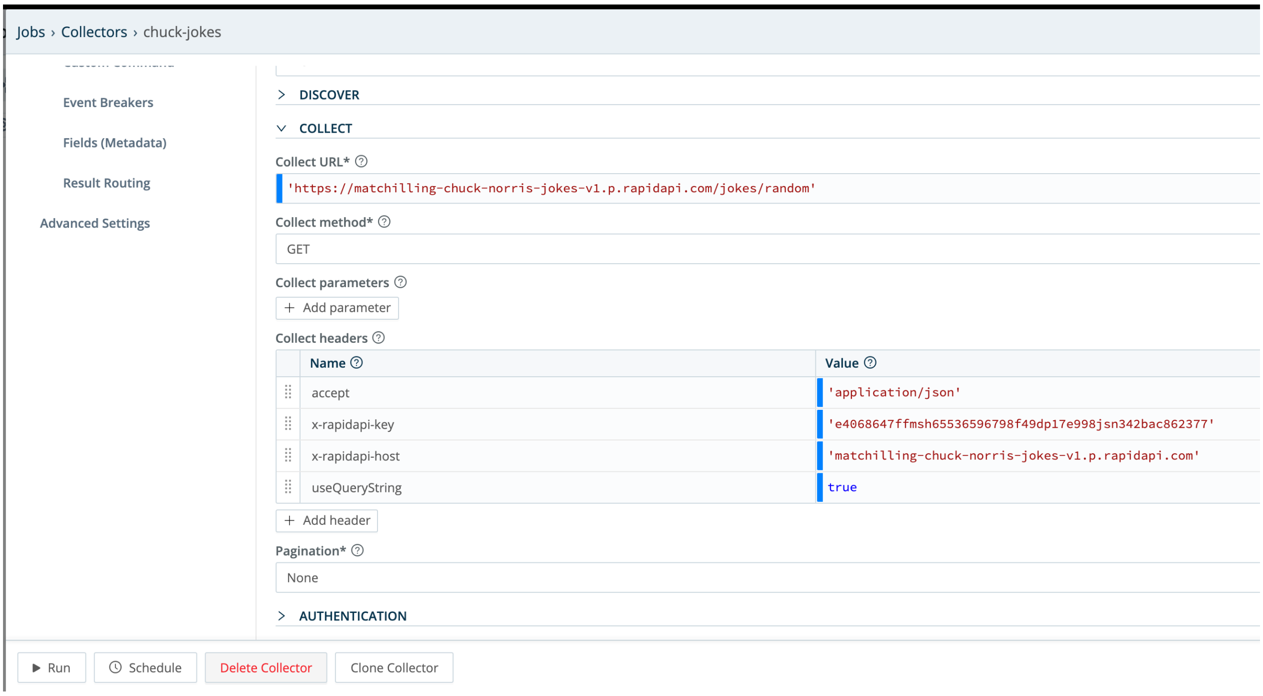
Collect headers:

```
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.



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

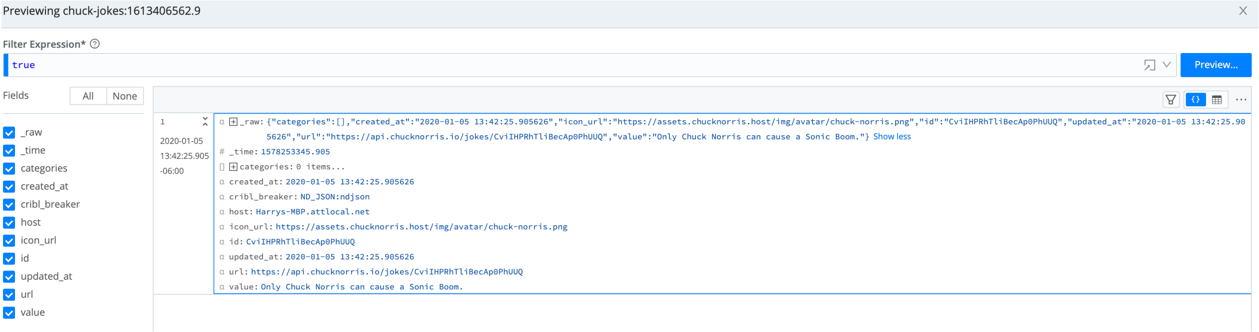
+ Add header

Pagination*
None

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:



Filter Expression*: true

```
_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 13:42:25.905626","url":"https://api.chucknorris.io/jokes/CviIHPRhTliBecAp0PhUUQ","value":"Only Chuck Norris can cause a Sonic Boom."} Show less
_time: 1578253345.905
categories: 0 items...
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 (i.e., 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 (i.e., 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.

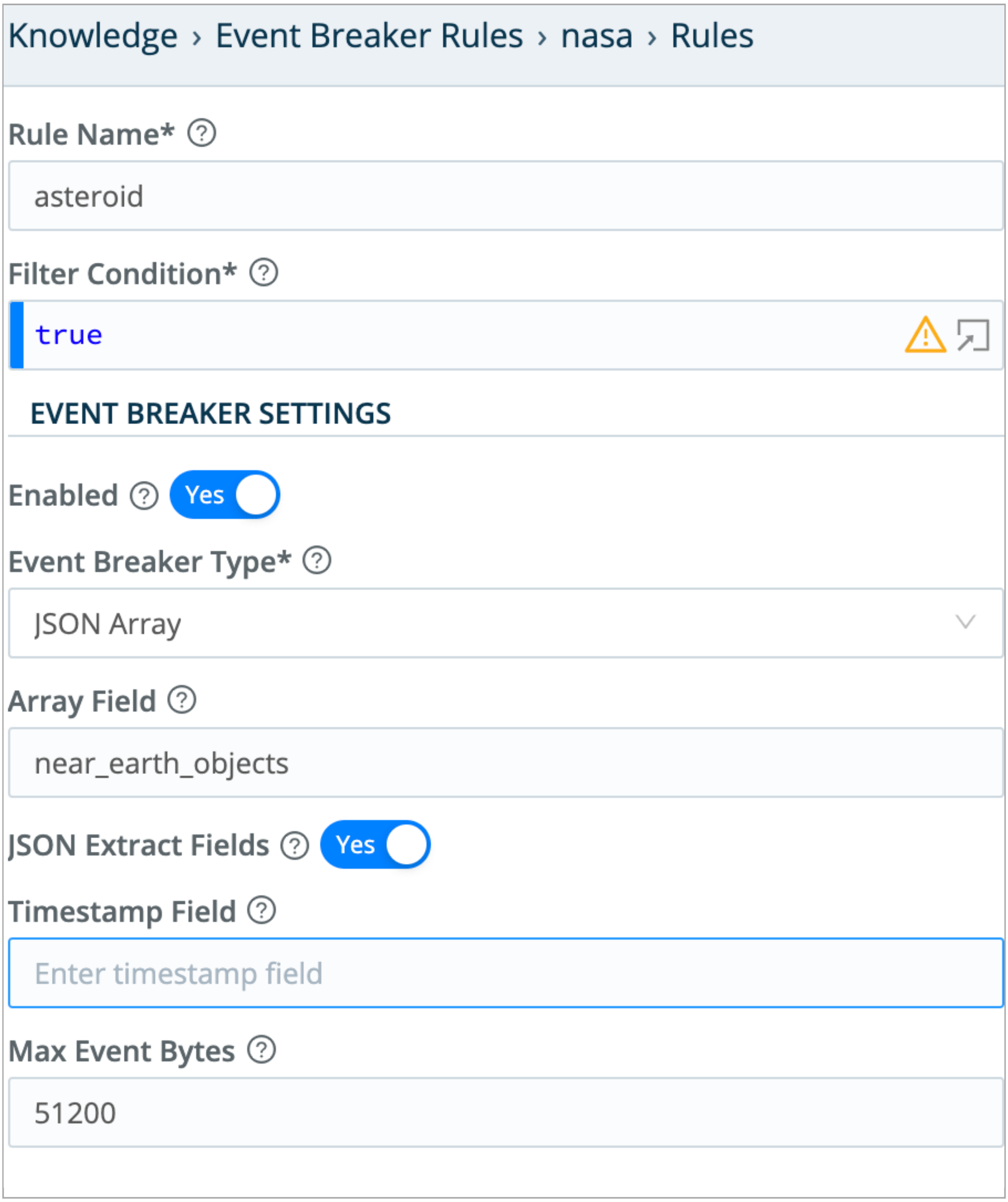
Collect headers: None

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

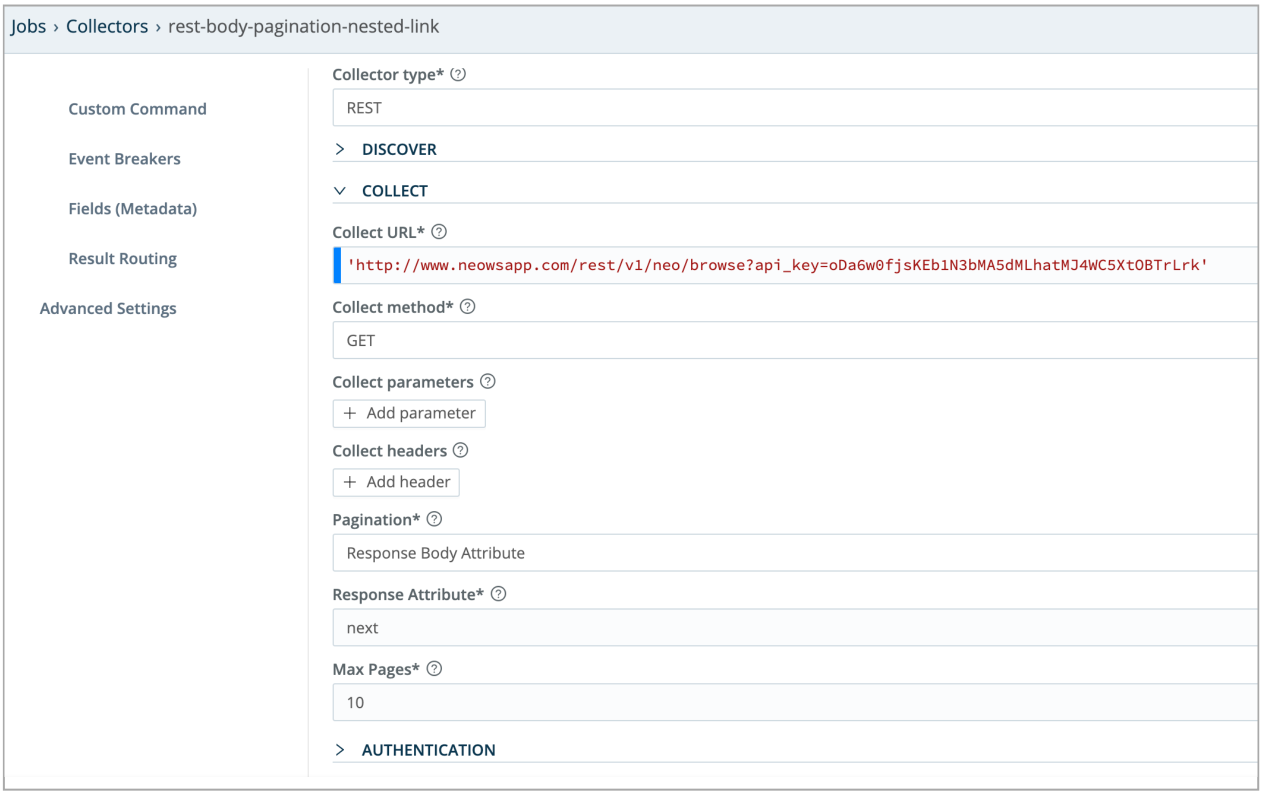
Timestamp Field ⓘ

Enter timestamp field

Max Event Bytes ⓘ

51200

Event Breaker configuration



Collector type*: REST

Collect URL*: `'http://www.neowsapp.com/rest/v1/neo/browse?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk'`

Collect method*: GET

Collect parameters

Collect headers

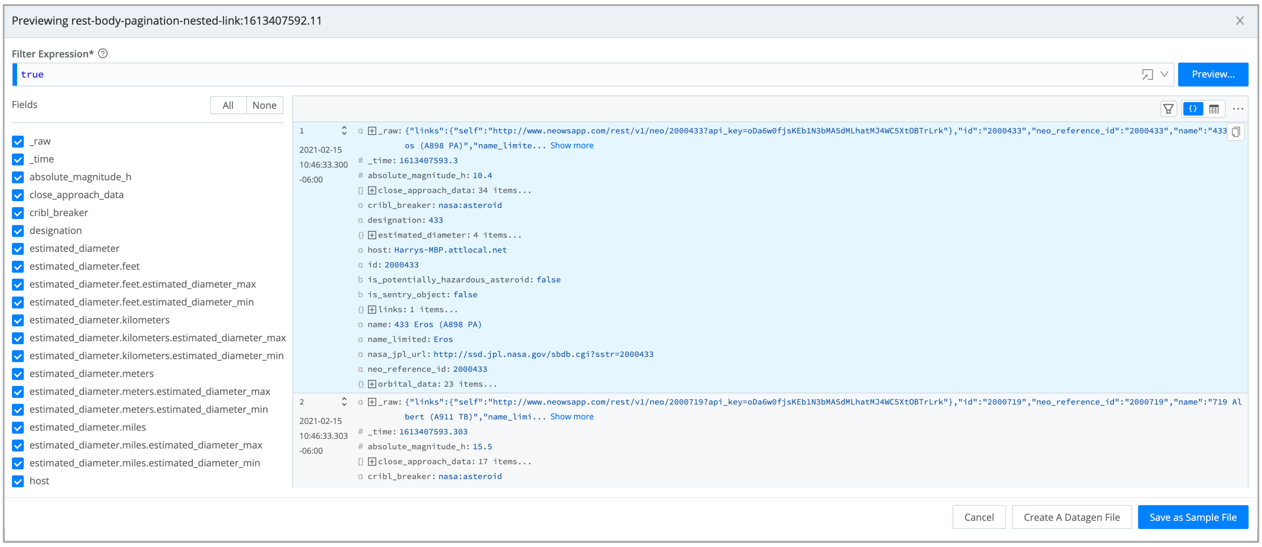
Pagination*: Response Body Attribute

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`

- ☑ _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>'
```

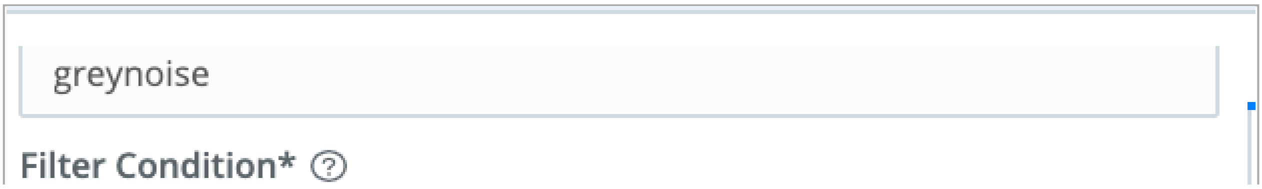
Pagination: Response Body Attribute

Response attribute: `scroll`

Max pages: `10` (or `0` to pull all data)

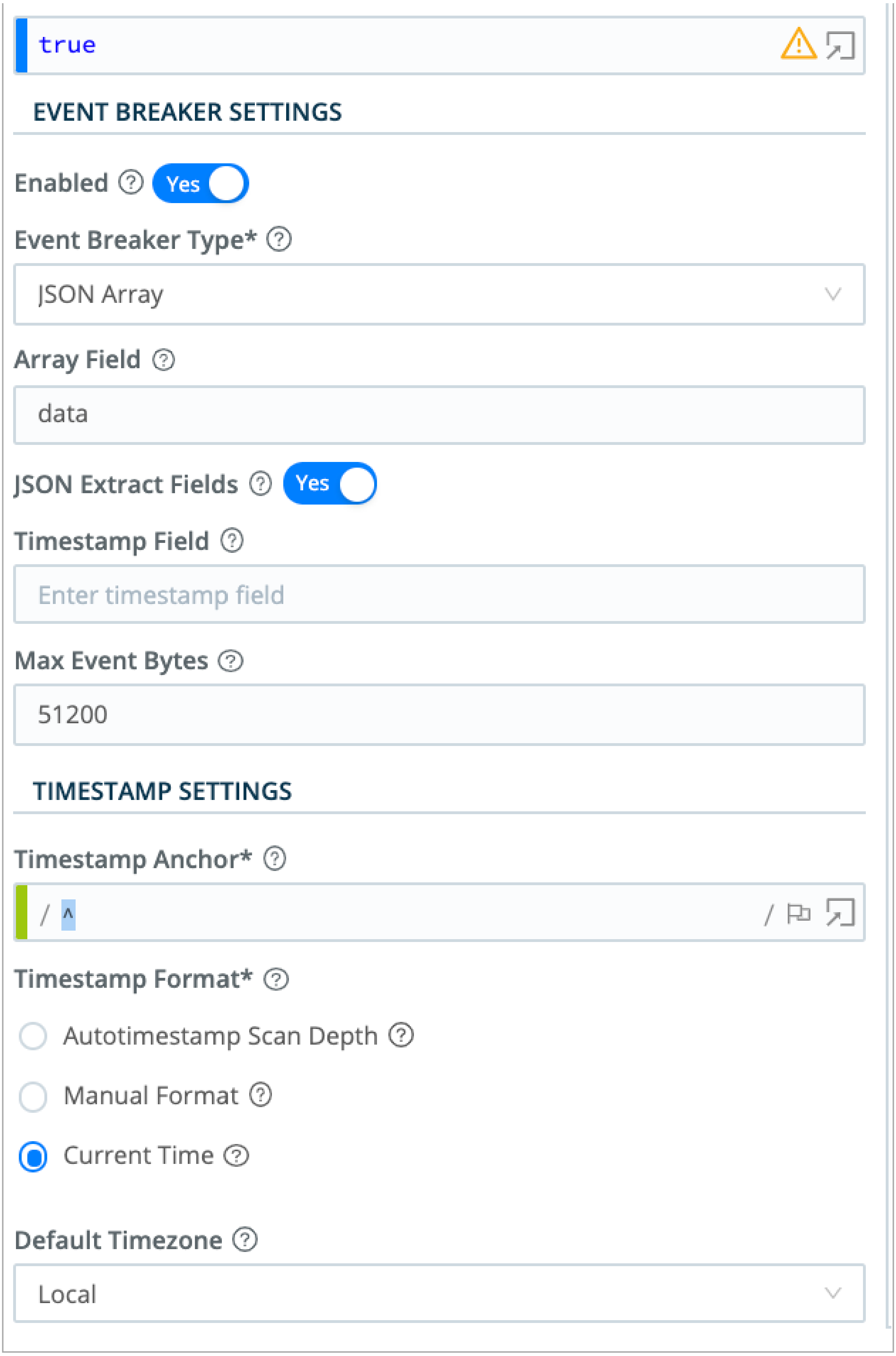
Authentication: None

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 ⓘ

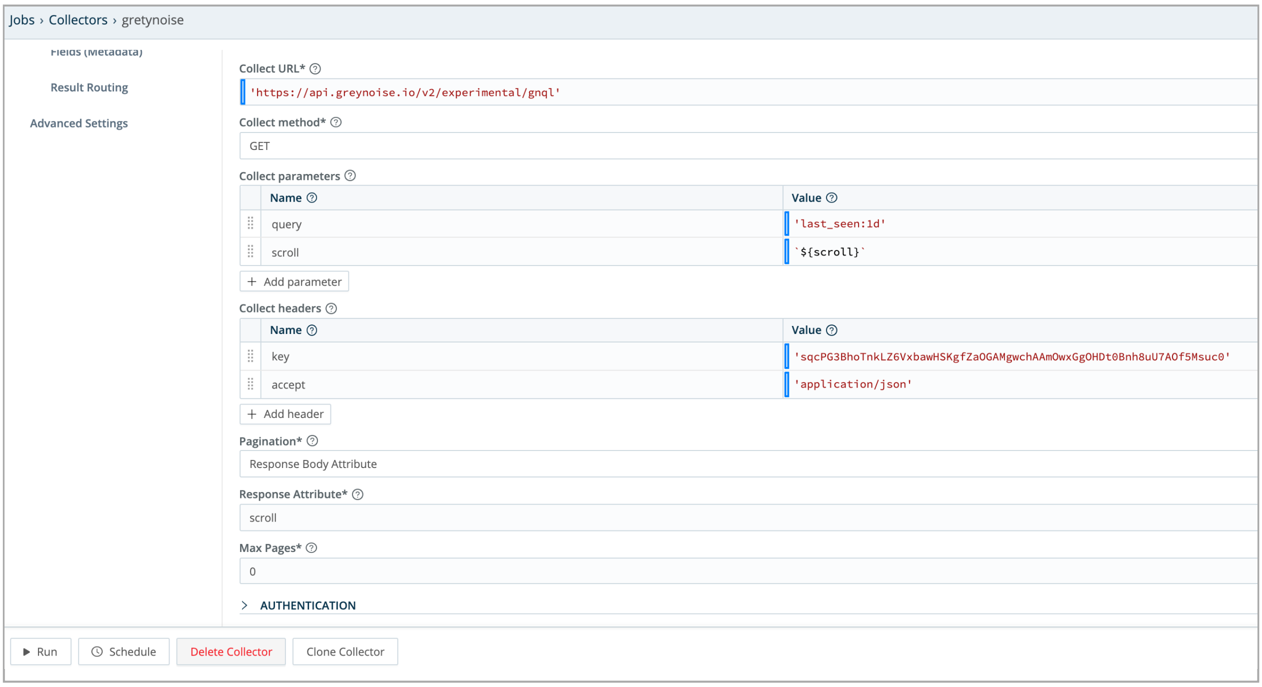
● Current Time ⓘ

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.



Jobs › Collectors › greytnoise

Fields (Metadata)
Result Routing
Advanced Settings

Collect URL*
'https://api.greynoise.io/v2/experimental/gnql'

Collect method*
GET

Collect parameters

Name	Value
query	'last_seen:1d'
scroll	`` `${scroll}` ``

+ Add parameter

Collect headers

Name	Value
key	'sqcPG3BhoTnkLZ6VxbawHSKgfZaOGAMgwchAAmOwxGgOHDt0Bnh8uU7AOf5Msuc0'
accept	'application/json'

+ Add header

Pagination*
Response Body Attribute

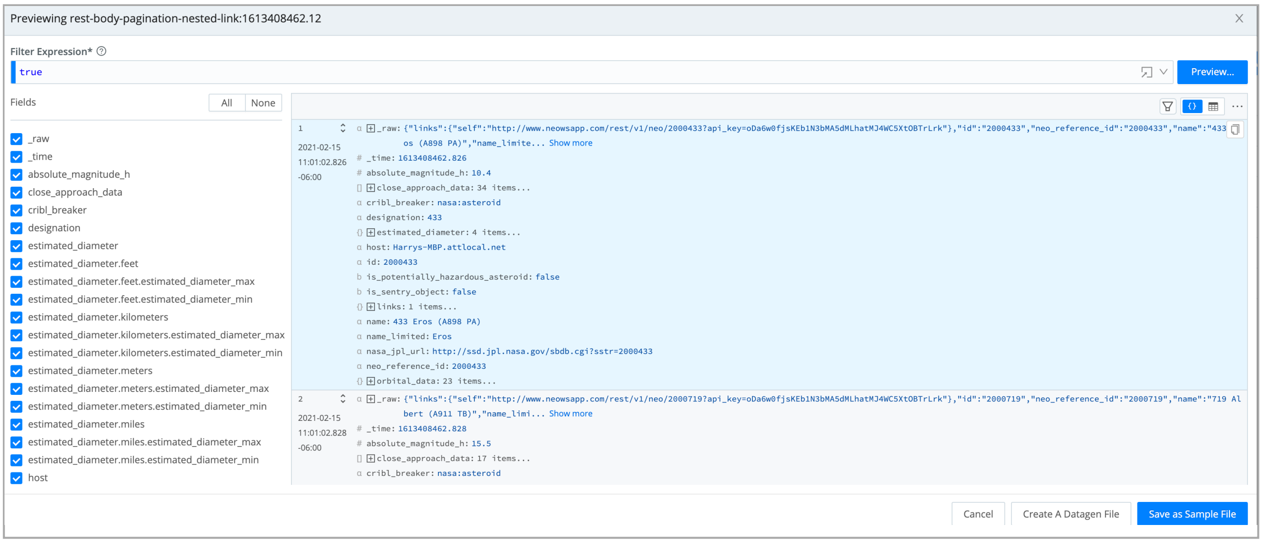
Response Attribute*
scroll

Max Pages*
0

› AUTHENTICATION

Collector configuration for HTTP GET, paginated via Response Body Attribute

Collector Output



Previewing rest-body-pagination-nested-link:1613408462.12

Filter Expression*
true

☑ _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            α ⊞_raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000433?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000433","neo_reference_id":"2000433","name":"433
2021-02-15              os (A898 PA)","name_limite... Show more
11:01:02.826 # _time: 1613408462.826
-06:00       # absolute_magnitude_h: 10.4
             [] ⊞close_approach_data: 34 items...
             α cribl_breaker: nasa:asteroid
             α designation: 433
             {} ⊞estimated_diameter: 4 items...
             α host: Harrys-MBP.attlocal.net
             α id: 2000433
             b is_potentially_hazardous_asteroid: false
             b is_sentry_object: false
             {} ⊞links: 1 items...
             α name: 433 Eros (A898 PA)
             α name_limited: Eros
             α nasa_jpl_url: http://ssd.jpl.nasa.gov/sbdb.cgi?sstr=2000433
             α neo_reference_id: 2000433
             {} ⊞orbital_data: 23 items...
2            α ⊞_raw: {"links":{"self":"http://www.neowsapp.com/rest/v1/neo/2000719?api_key=oDa6w0fjsKEb1N3bMA5dMLhatMJ4WC5XtOBTrLrk"},"id":"2000719","neo_reference_id":"2000719","name":"719 Al
2021-02-15              bert (A911 TB)","name_limi... Show more
11:01:02.828 # _time: 1613408462.828
-06:00       # absolute_magnitude_h: 15.5
             [] ⊞close_approach_data: 17 items...
             α cribl_breaker: nasa:asteroid
```

Paginated events

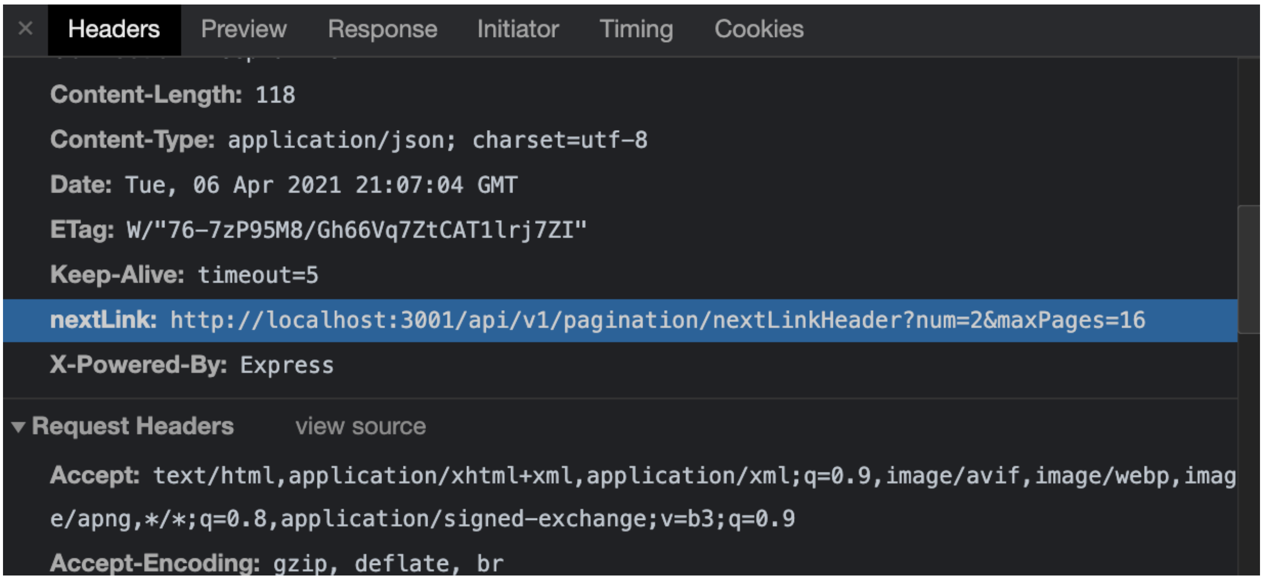
> This API allows a certain number of calls/month. Cribl recommends that you not schedule this Collector – run it ad-hoc, for testing only.

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

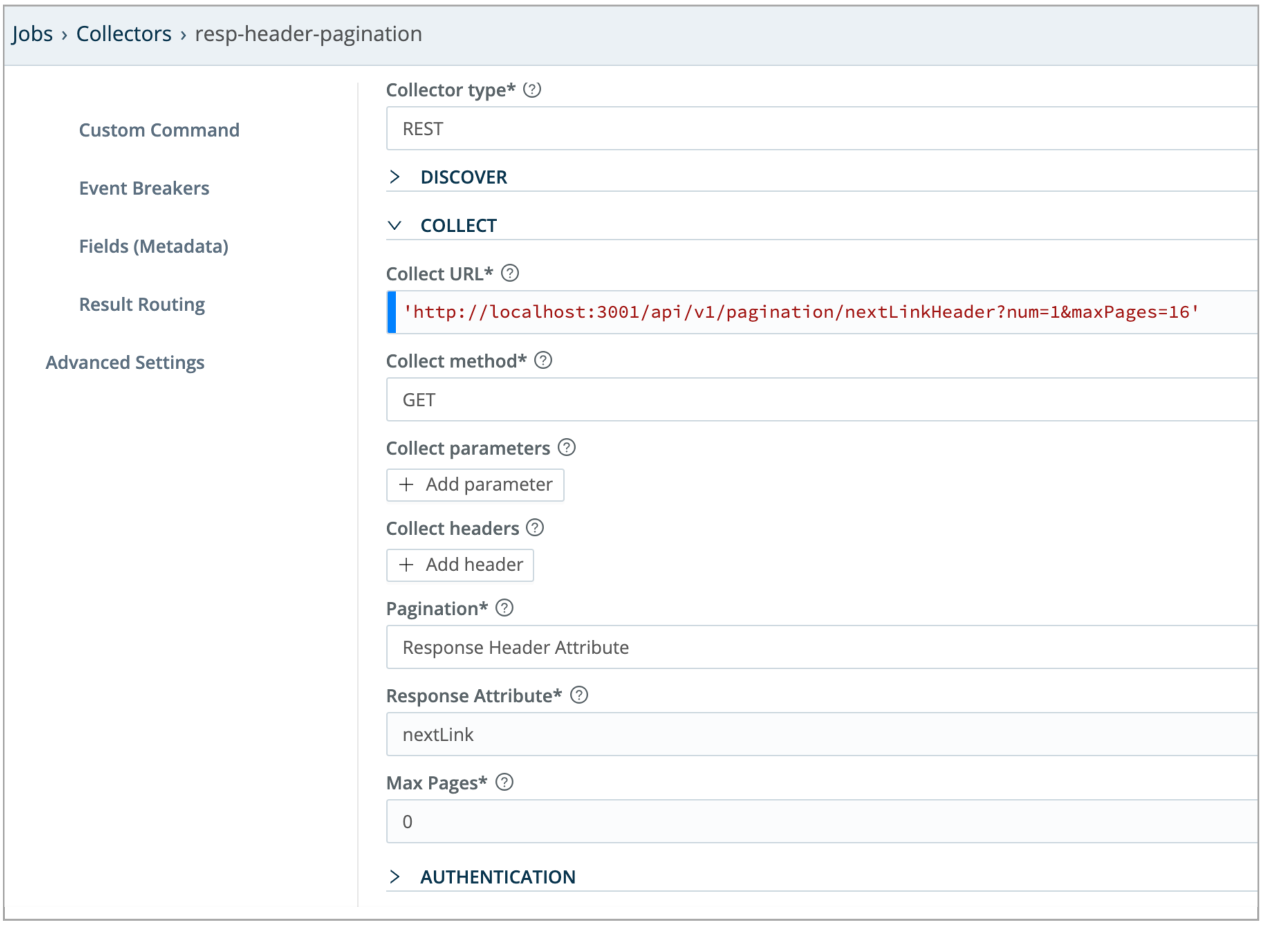
Pagination: Response Header Attribute

Response attribute: `nextLink`

Authentication: None

Event Breaker: None

> You can modify the `maxPages` URL parameter to control how many pages this call returns.



Collector type*: REST

Collect URL*: `'http://localhost:3001/api/v1/pagination/nextLinkHeader?num=1&maxPages=16'`

Collect method*: GET

Collect parameters

Collect headers

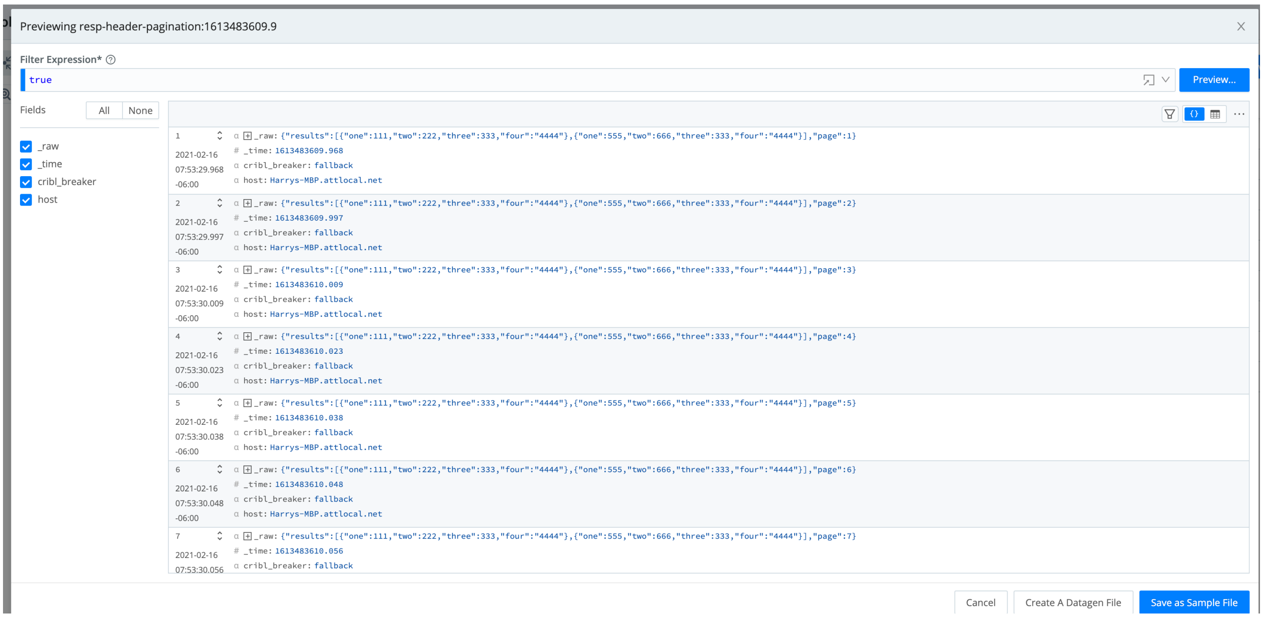
Pagination*: Response Header Attribute

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`

☑ _raw
☑ _time
☑ cribl_breaker
☑ host

```
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 (e.g., 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 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)

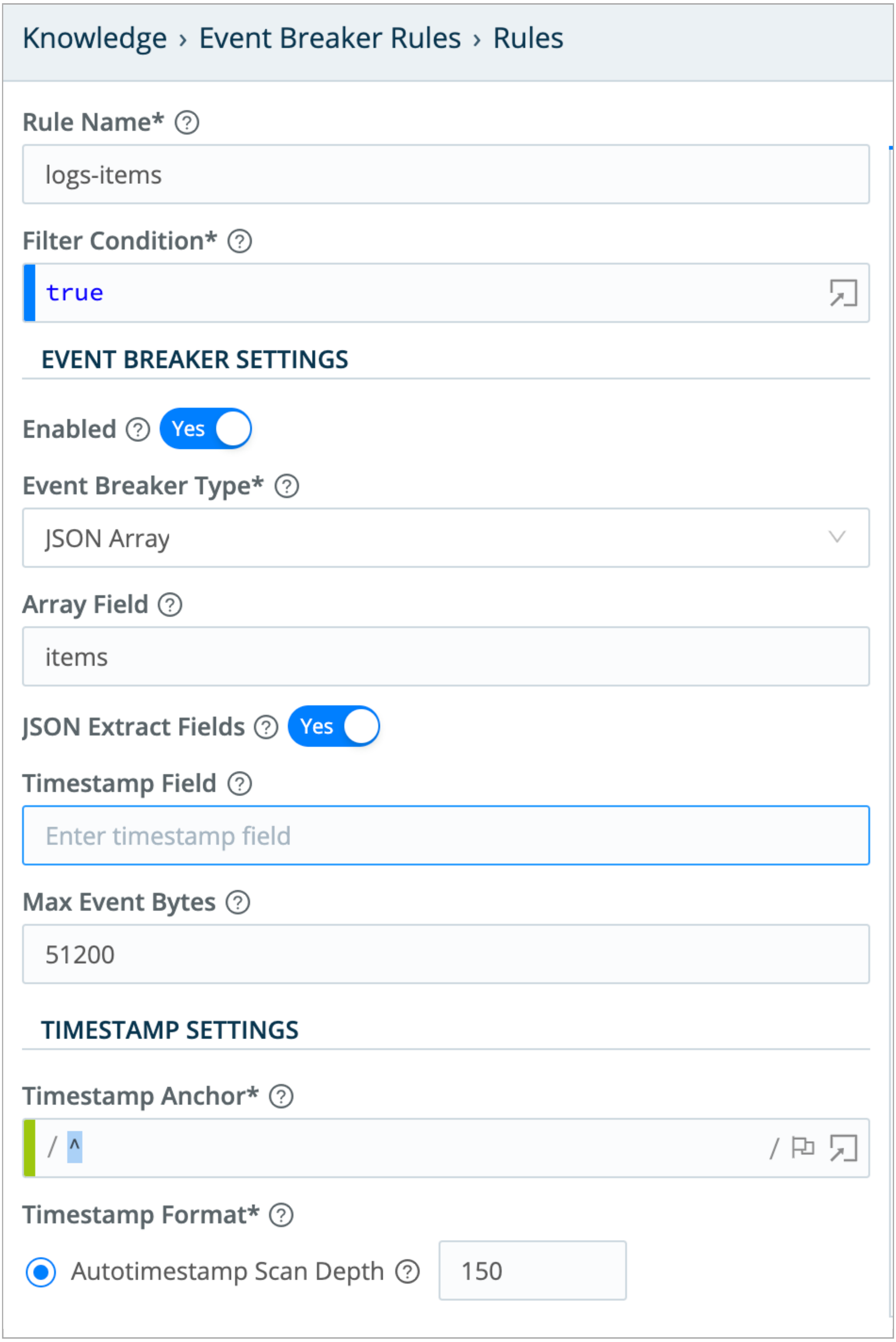
[Authentication] POST Body: `` `{ "username": "${username}", "password": "${password}" }` ``

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

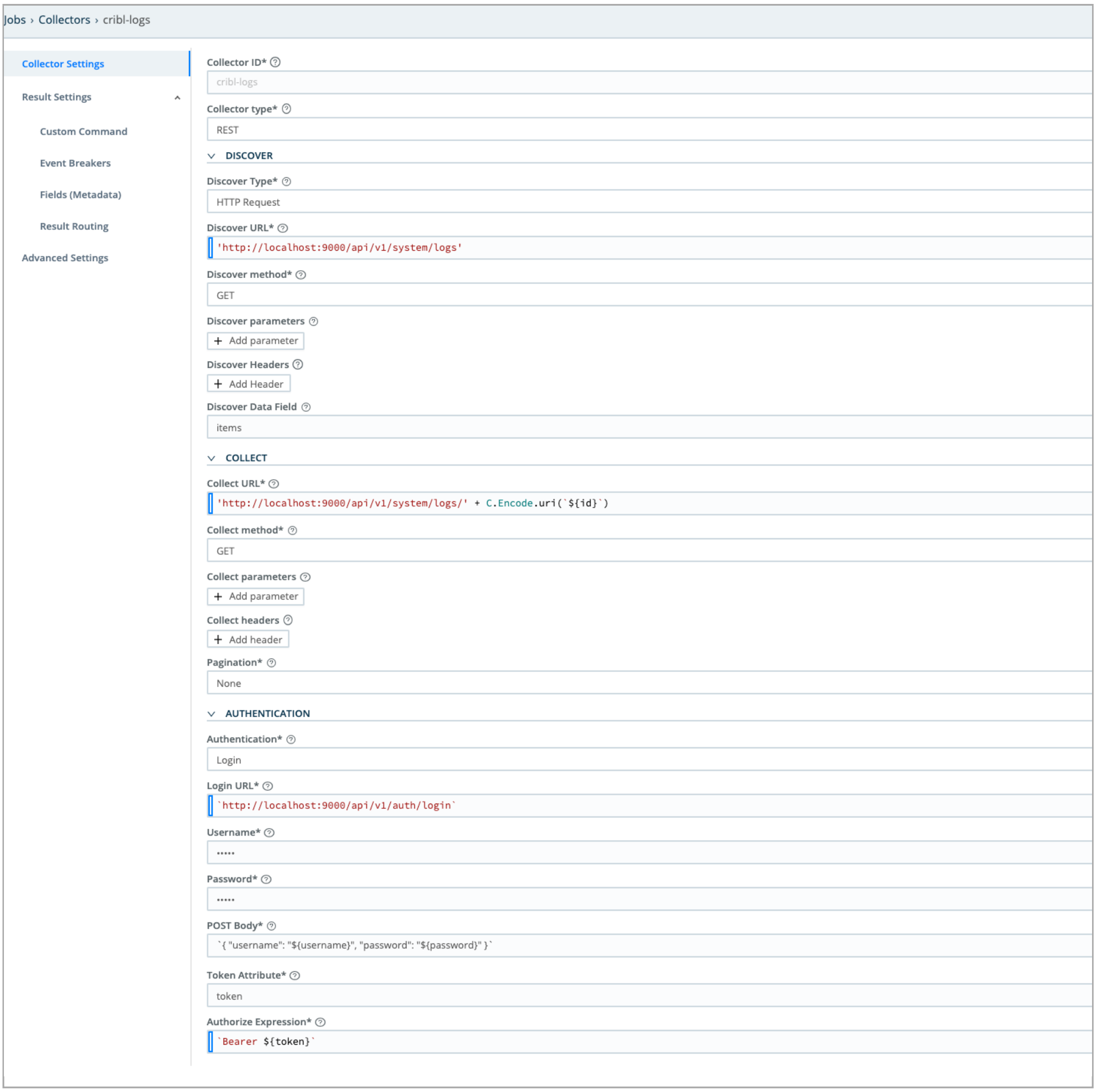
Timestamp Anchor* ⓘ

/ ^ /

Timestamp Format* ⓘ

● Autotimestamp Scan Depth ⓘ 150

Event Breaker configuration



Collector ID*: cribl-logs

Collector type*: REST

DISCOVER

Discover Type*: HTTP Request

Discover URL*: `'http://localhost:9000/api/v1/system/logs'`

Discover method*: GET

Discover parameters

Discover Headers

Discover Data Field: items

COLLECT

Collect URL*: `'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)`

Collect method*: GET

Collect parameters

Collect headers

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

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, Discover and Collect). The variable name used in the **Authorize Expression** should be the same name specified in the **Token Attribute** call.

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"
    }
  ]
}
```

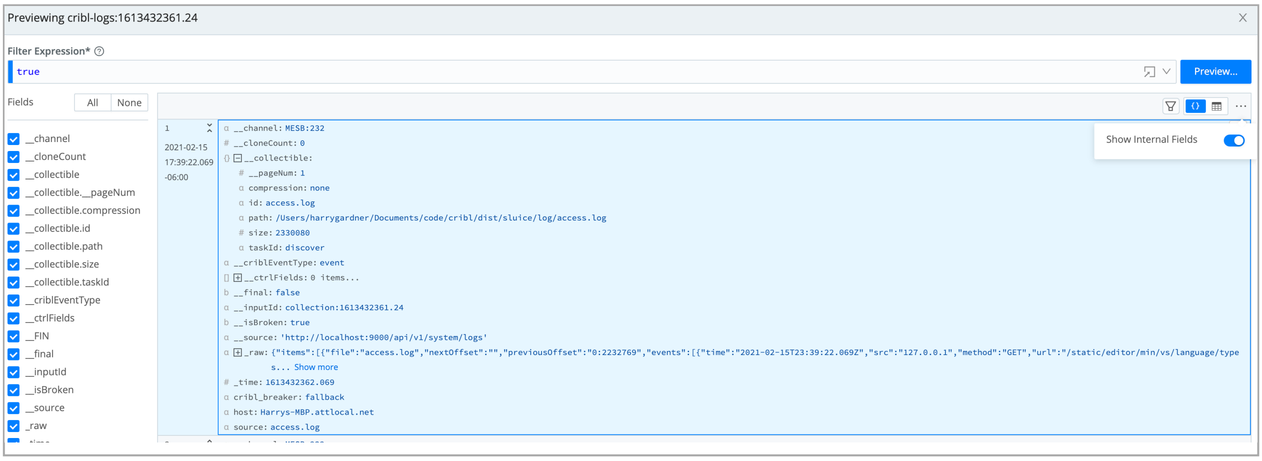
The **Discover Data Field** is used to define the array in Discover results that contains the list of items to discover. Here, each item is an object, with an attribute ID that is referenced in the Collect calls. So the Discover call generates a list of items for which Collect tasks will be created.

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:



Filter Expression*: true

Fields: ☑ __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

__channel: MESB:232
__cloneCount: 0
__collectible:
  __pageNum: 1
  compression: none
  id: access.log
  path: /Users/harrygardner/Documents/code/cribl/dist/sluice/log/access.log
  size: 2330080
  taskId: discover
__criblEventType: event
__ctrlFields: 0 items...
__final: false
__inputId: collection:1613432361.24
__isBroken: true
__source: 'http://localhost:9000/api/v1/system/logs'
_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
cribl_breaker: fallback
host: Harrys-MBP.attlocal.net
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:

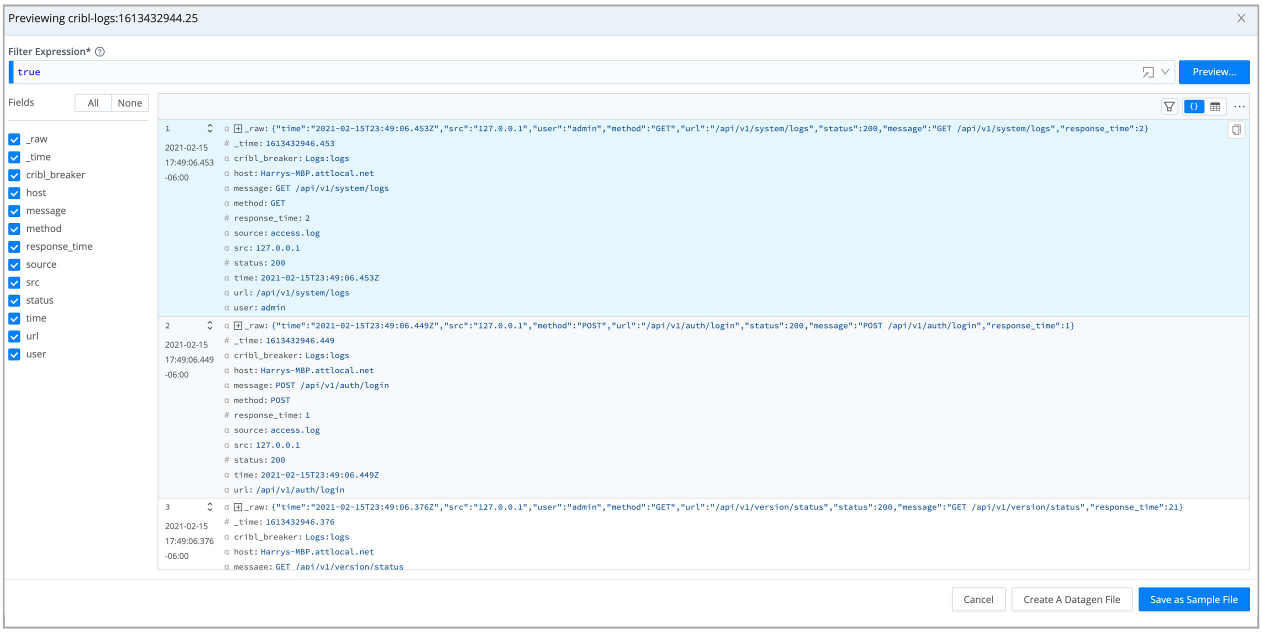
```
'http://localhost:9000/api/v1/system/logs/' + C.Encode.uri(`${id}`)
```

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:



Previewing cribl-logs:1613432944.25

Filter Expression*: `true`

Fields: All | None

☑ _raw
☑ _time
☑ cribl_breaker
☑ host
☑ message
☑ method
☑ response_time
☑ source
☑ src
☑ status
☑ time
☑ url
☑ user

```
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
```

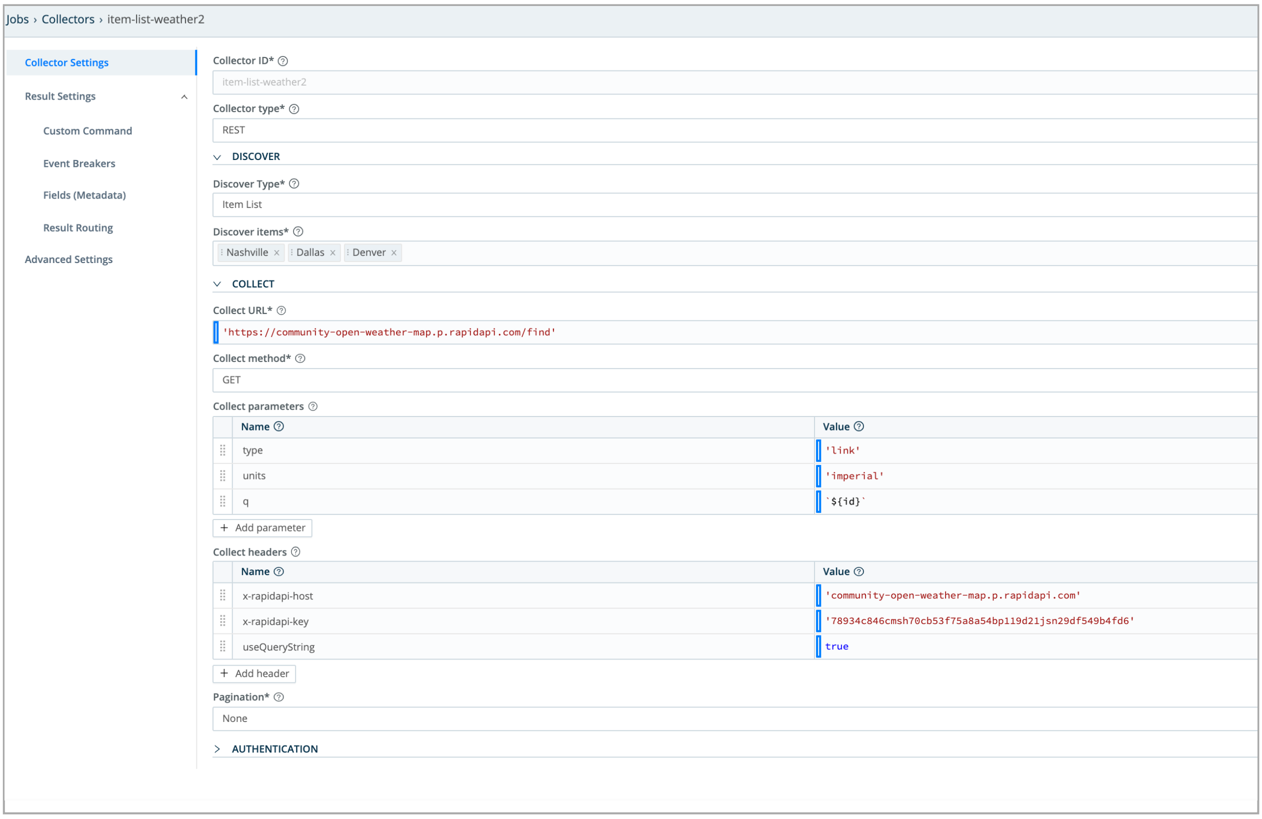
Pagination: None

Authentication: None

Event Breaker: JSON Newline Delimited – Use a rule like **Cribl > ndjson** to parse each event and extract fields.

Fields (Metadata):

```
job: weather-${__collectible.id}

city: ${__collectible.id}
```



Collector ID*: item-list-weather2

Collector type*: REST

DISCOVER

Discover Type*: Item List

Discover items*: Nashville × Dallas × Denver ×

COLLECT

Collect URL*: 'https://community-open-weather-map.p.rapidapi.com/find'

Collect method*: GET

Collect parameters

Name	Value
type	'link'
units	'imperial'
q	\`${id}\`

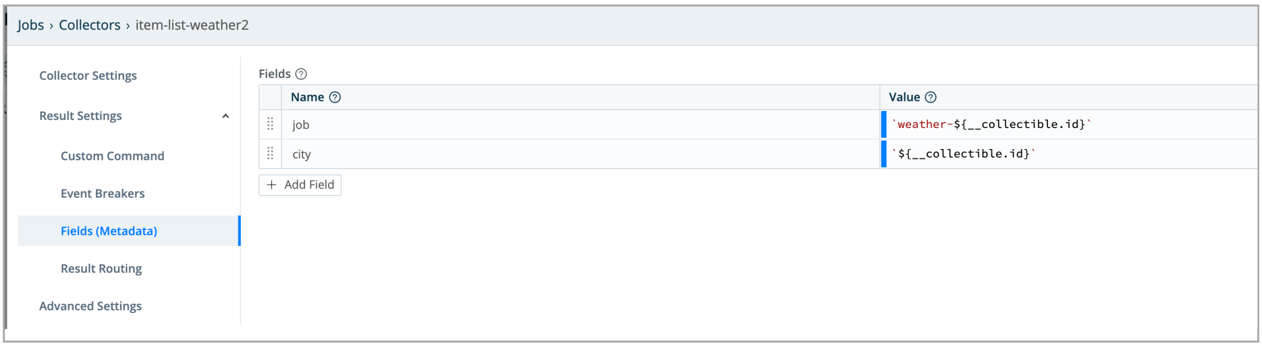
Collect headers

Name	Value
x-rapidapi-host	'community-open-weather-map.p.rapidapi.com'
x-rapidapi-key	'78934c846cmsh70cb53f75a8a54bp119d21jsn29df549b4fd6'
useQueryString	true

Pagination*: None

AUTHENTICATION

Collector configuration for Discovery via Item List



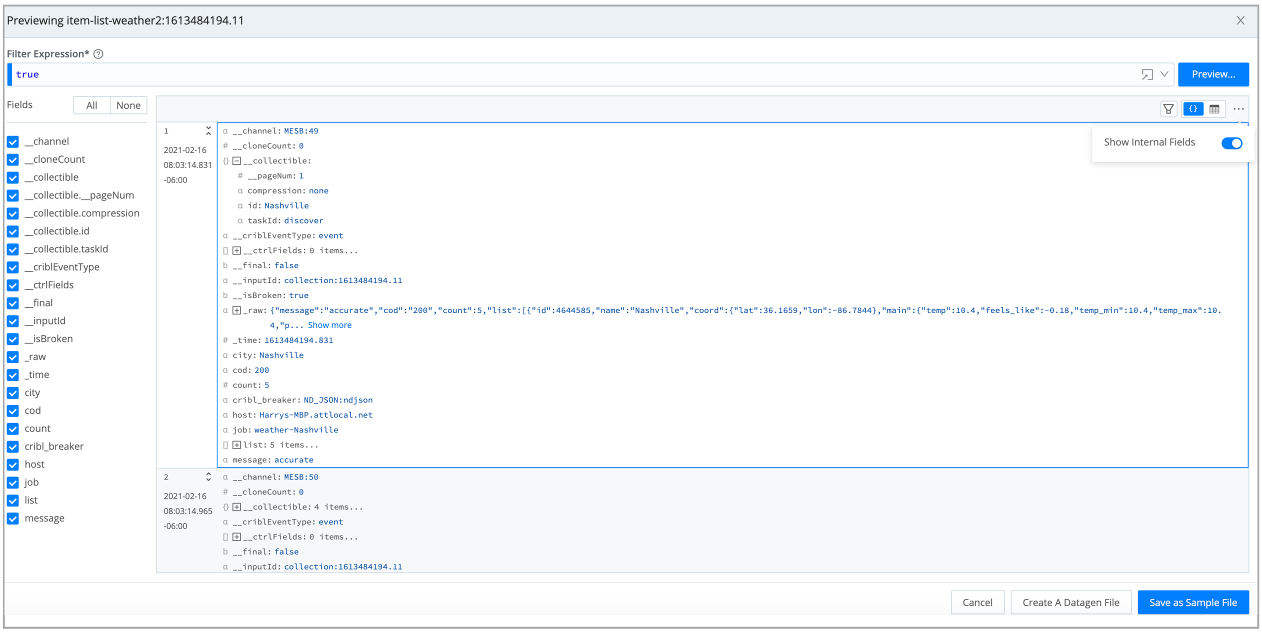
Name	Value
job	`` `weather-${__collectible.id}` ``
city	`` `${__collectible.id}` ``

Fields (Metadata) configuration

Collector Output

One interesting thing about this example is the addition of **Fields (Metadata)** 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:



```
true
```

```
1   2021-02-16 08:03:14.831 -06:00
α __channel: MESB:49
# __cloneCount: 0
{} ⊟ __collectible:
    # __pageNum: 1
    α compression: none
    α id: Nashville
    α taskId: discover
α __criblEventType: event
[] ⊞ __ctrlFields: 0 items...
b __final: false
α __inputId: collection:1613484194.11
b __isBroken: true
α ⊞ _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
α city: Nashville
α cod: 200
# count: 5
α cribl_breaker: ND_JSON:ndjson
α host: Harrys-MBP.attlocal.net
α job: weather-Nashville
[] ⊞ list: 5 items...
α message: accurate

2   2021-02-16 08:03:14.965 -06:00
α __channel: MESB:50
# __cloneCount: 0
{} ⊞ __collectible: 4 items...
α __criblEventType: event
[] ⊞ __ctrlFields: 0 items...
b __final: false
α __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

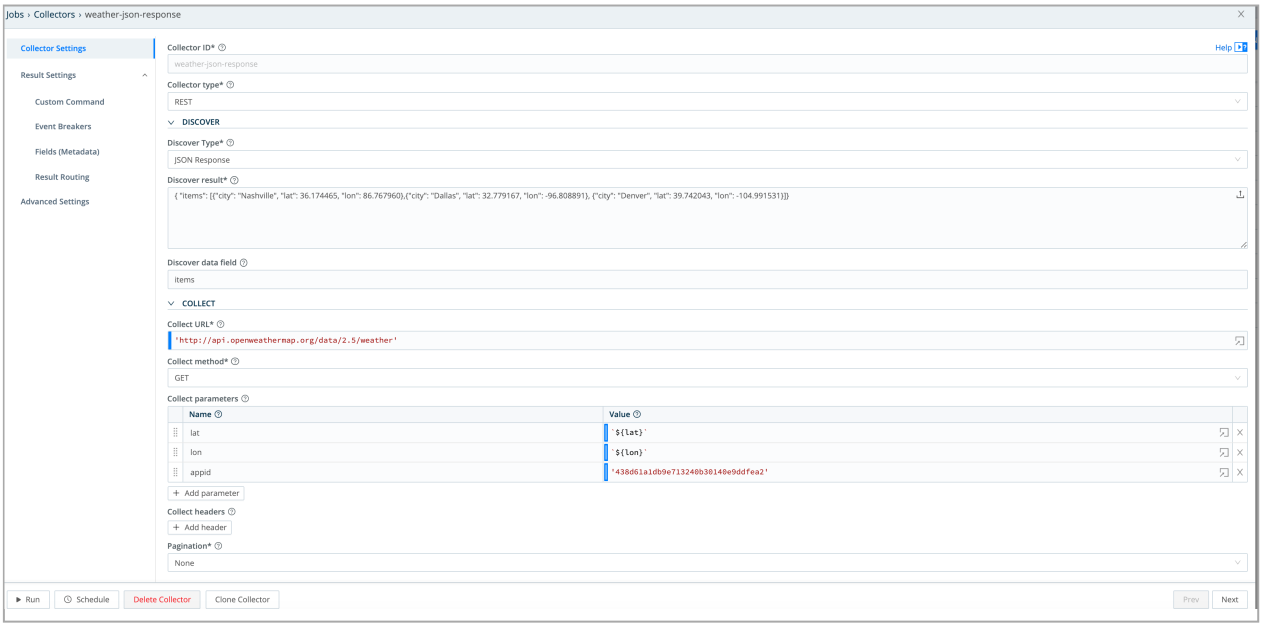
Authentication: None

Event Breaker: JSON Newline Delimited – Use a rule like **Cribl > ndjson** to parse each event and extract fields.

Fields (Metadata):

```
job: `weather-${__collectible.city}`
```

```
city: `${__collectible.city}`
```



Collector ID*: weather-json-response

Collector type*: REST

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'`

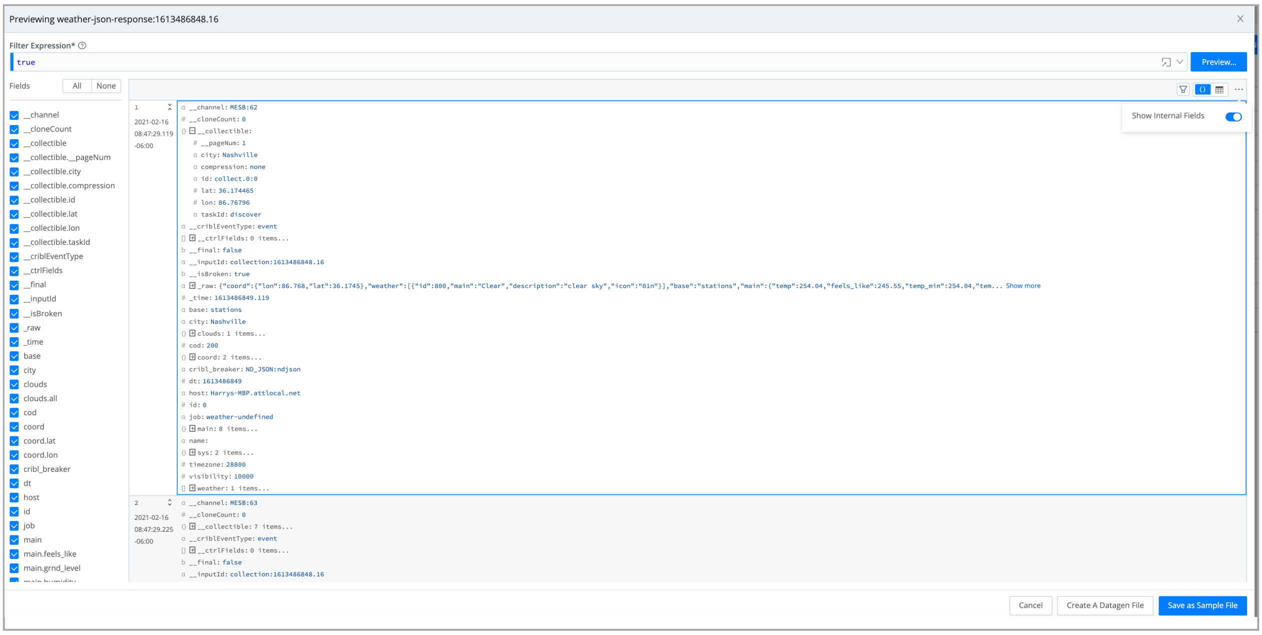
Collect headers

Pagination*: None

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
α __channel: MESB:62
# __cloneCount: 0
{} ⊟ __collectible:
    # __pageNum: 1
    α city: Nashville
    α compression: none
    α id: collect.0:0
    # lat: 36.174465
    # lon: 86.76796
    α taskId: discover
α __criblEventType: event
[] ⊞ __ctrlFields: 0 items...
b __final: false
α __inputId: collection:1613486848.16
b __isBroken: true
α ⊞ _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
α base: stations
α city: Nashville
{} ⊞ clouds: 1 items...
# cod: 200
{} ⊞ coord: 2 items...
α cribl_breaker: ND_JSON:ndjson
# dt: 1613486849
α host: Harrys-MBP.attlocal.net
# id: 0
α job: weather-undefined
{} ⊞ main: 8 items...
α name:
{} ⊞ sys: 2 items...
# timezone: 28800
# visibility: 10000
[] ⊞ weather: 1 items...

2
2021-02-16
08:47:29.225
-06:00
α __channel: MESB:63
# __cloneCount: 0
{} ⊞ __collectible: 7 items...
α __criblEventType: event
[] ⊞ __ctrlFields: 0 items...
b __final: false
α __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.

;

11.6.3. 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 New** 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.

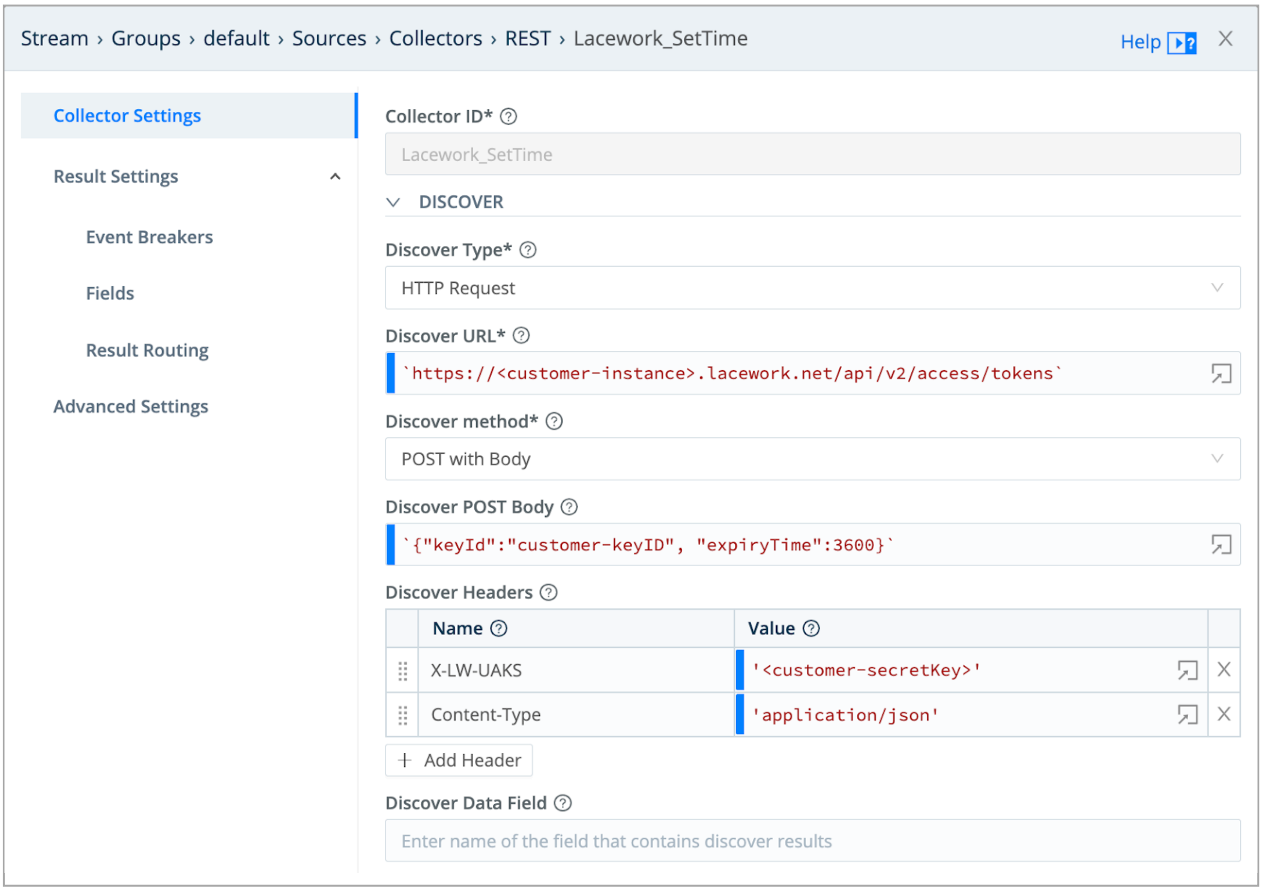
Discover method: From the drop-down, select `POST with Body`.

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:



Collector ID*: Lacework_SetTime

Discover Type*: HTTP Request

Discover URL*: `` `https://<customer-instance>.lacework.net/api/v2/access/tokens` ``

Discover method*: POST with Body

Discover POST Body: `` `{"keyId":"customer-keyID", "expiryTime":3600}` ``

Discover Headers:

Name	Value
X-LW-UAKS	`'<customer-secretKey>'`
Content-Type	`'application/json'`

Discover Data Field: Enter name of the field that contains discover results

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": "${C.Time.strftime(earliest || new
Date().getTime()/1000-(24*60*60), '%Y-%m-%dT%H:%M:%SZ')}", "endTime":
"${C.Time.strftime(latest || new Date().getTime()/1000, '%Y-%m-
%dT%H:%M:%SZ')}"},"filters": [ { "expression":"eq","field":"vulnId", "value": "CVE-
2018-11233" } ] }`
```

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.

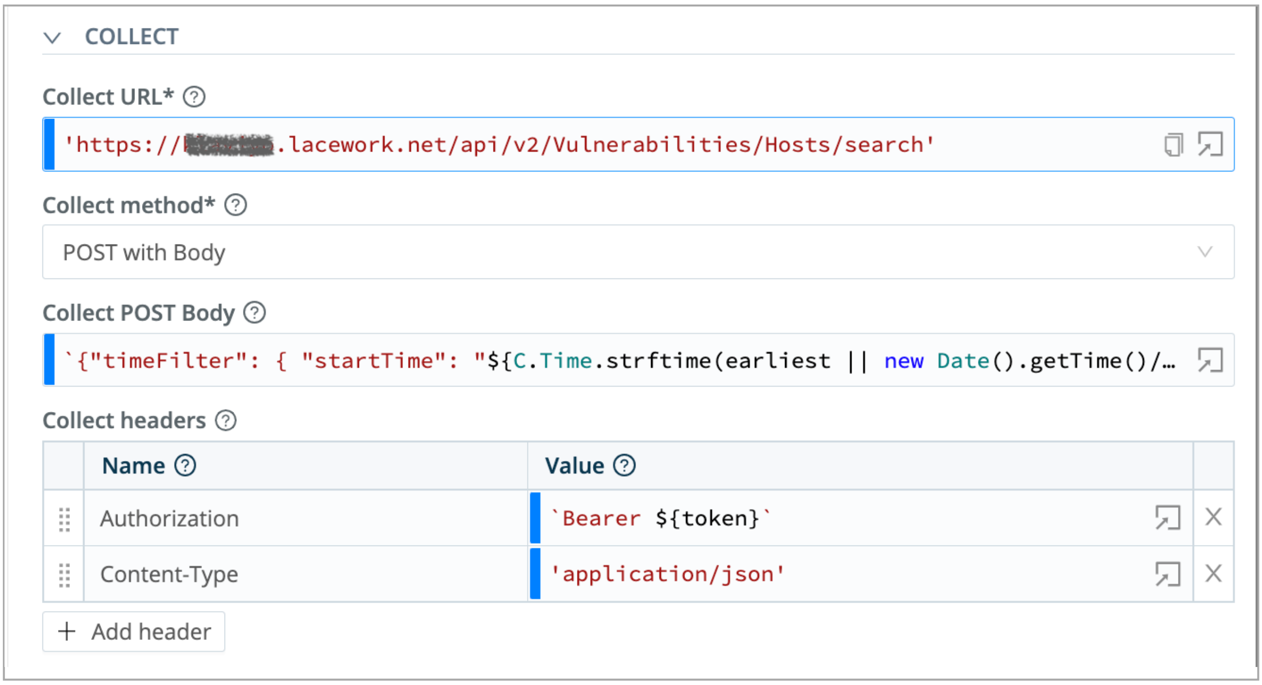
Collect Headers

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://[redacted].lacework.net/api/v2/Vulnerabilities/Hosts/search'
```

Collect method*

POST with Body

Collect POST Body

```
`{"timeFilter": { "startTime": "${C.Time.strftime(earliest || new Date().getTime()/…
```

Collect headers

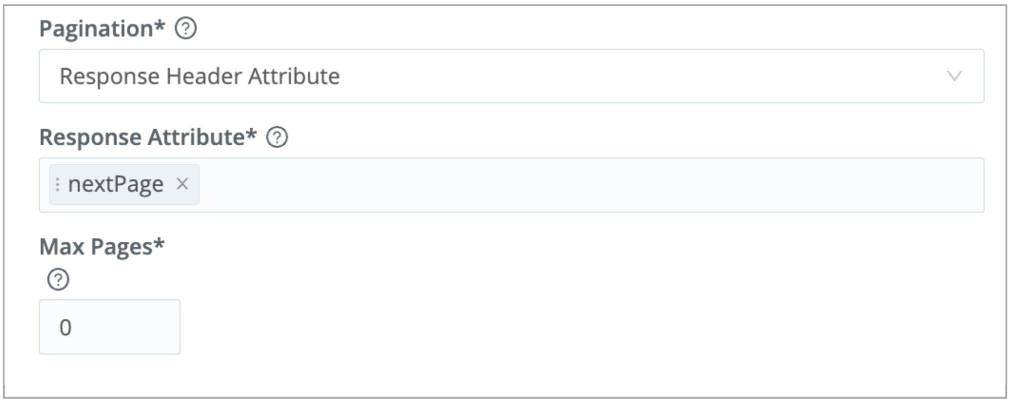
Name	Value
Authorization	`` `Bearer ${token}` ``
Content-Type	`'application/json'`

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

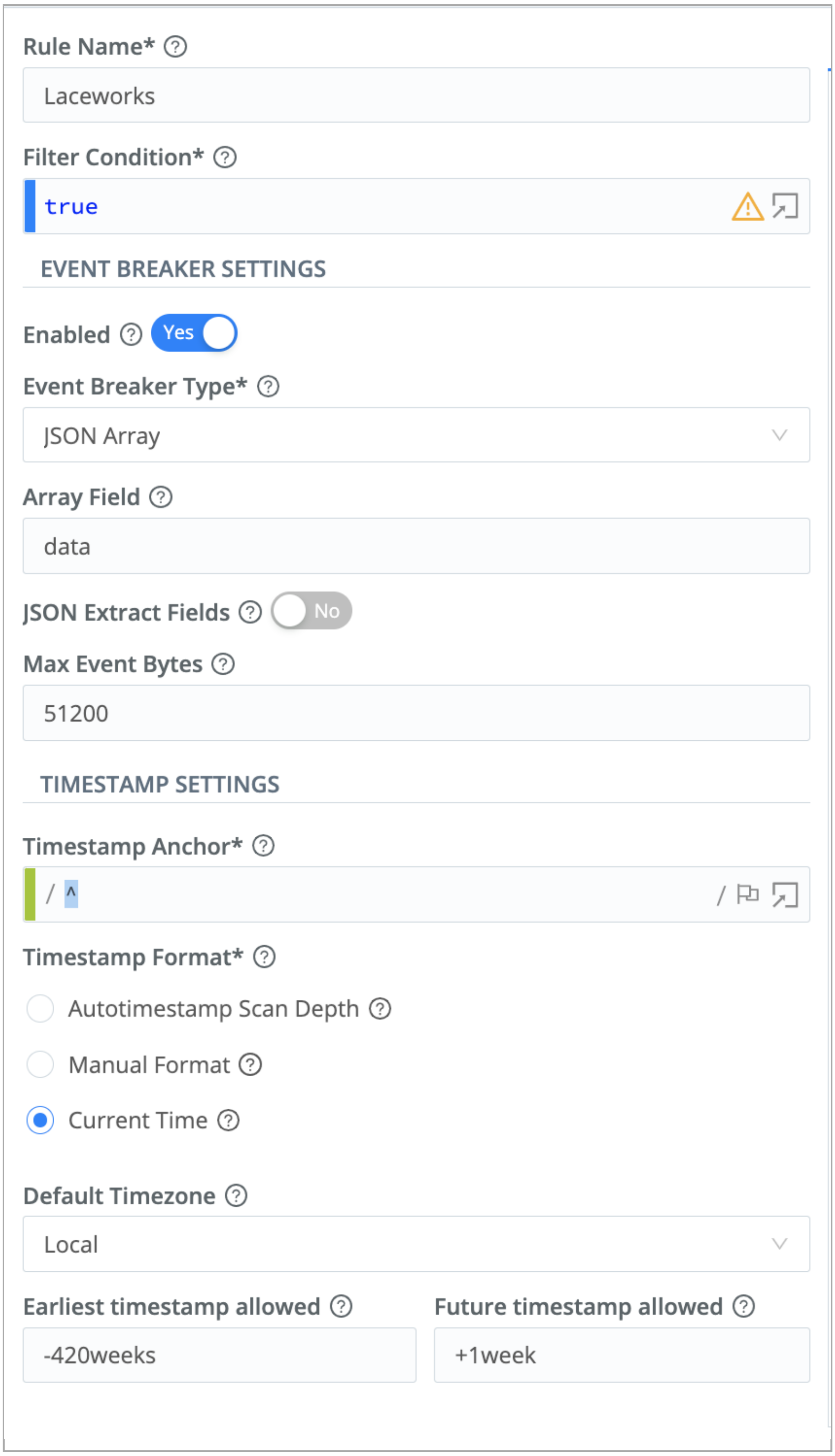
Every call to the Lacework API `v2/Vulnerabilities/Hosts/search` endpoint returns data formatted as a single event. Every event contains multiple nested JSON arrays, where each array is a `data` element. We'll now create an Event Breaker that parses the larger structure into individual events – one for 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 **+ Add New** 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

Creating a ruleset

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

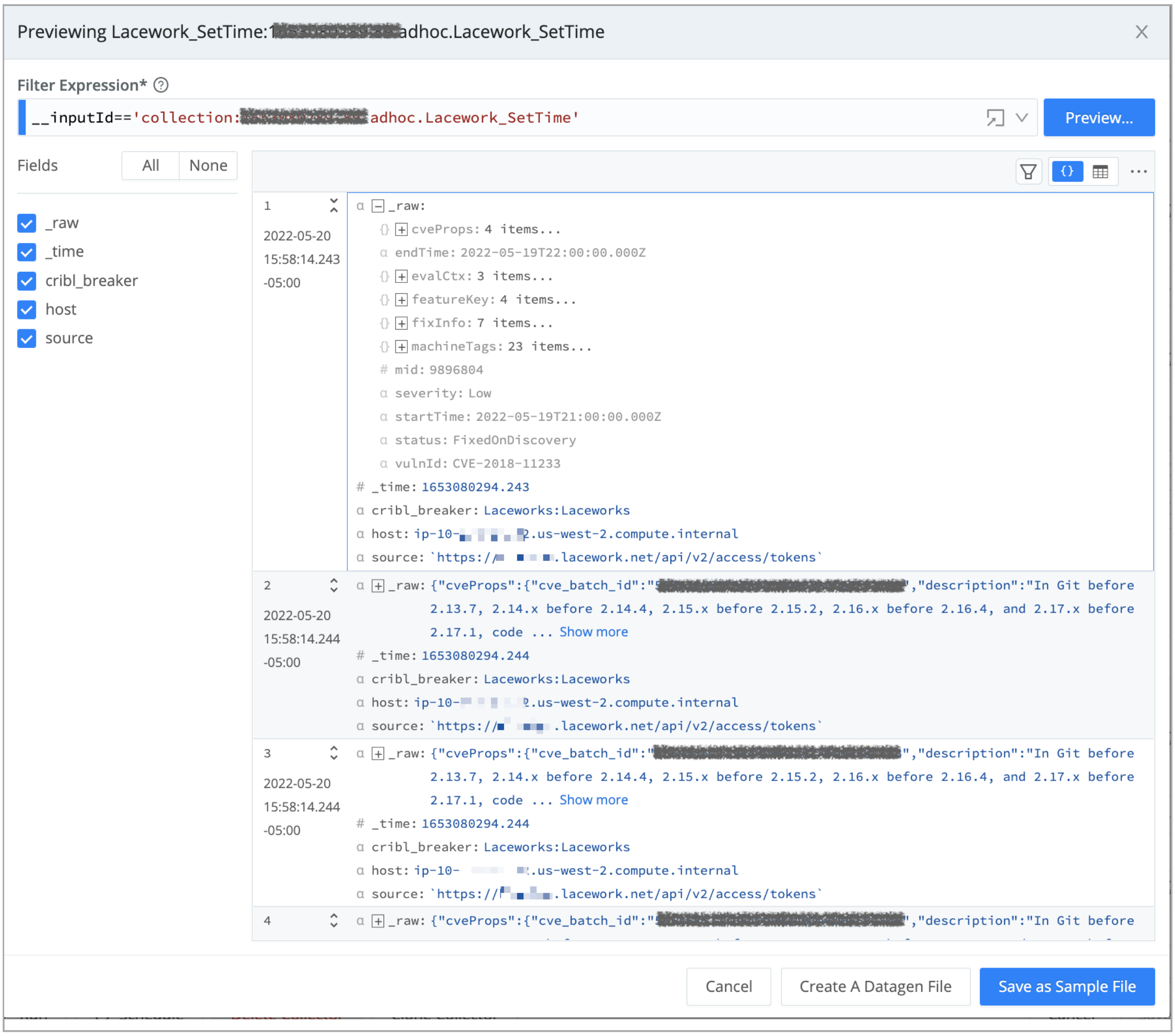
We'll start with the Discovery run:

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:



Previewing Lacework_SetTime:[redacted]adhoc.Lacework_SetTime

Filter Expression*

```
__inputId=='collection:[redacted]adhoc.Lacework_SetTime'
```

☑ _raw
☑ _time
☑ cribl_breaker
☑ host
☑ source

```
1     2022-05-20 15:58:14.243 -05:00
      α ⊟ _raw:
          {} ⊞ cveProps: 4 items...
          α endTime: 2022-05-19T22:00:00.000Z
          {} ⊞ evalCtx: 3 items...
          {} ⊞ featureKey: 4 items...
          {} ⊞ fixInfo: 7 items...
          {} ⊞ machineTags: 23 items...
          # mid: 9896804
          α severity: Low
          α startTime: 2022-05-19T21:00:00.000Z
          α status: FixedOnDiscovery
          α vulnId: CVE-2018-11233
      # _time: 1653080294.243
      α cribl_breaker: Laceworks:Laceworks
      α host: ip-10-[redacted]2.us-west-2.compute.internal
      α source: `https:/[redacted].lacework.net/api/v2/access/tokens`

2     2022-05-20 15:58:14.244 -05:00
      α ⊞ _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
      α cribl_breaker: Laceworks:Laceworks
      α host: ip-10-[redacted]2.us-west-2.compute.internal
      α source: `https:/[redacted].lacework.net/api/v2/access/tokens`

3     2022-05-20 15:58:14.244 -05:00
      α ⊞ _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
      α cribl_breaker: Laceworks:Laceworks
      α host: ip-10-[redacted].us-west-2.compute.internal
      α source: `https://[redacted].lacework.net/api/v2/access/tokens`

4     α ⊞ _raw: {"cveProps":{"cve_batch_id":"[redacted]","description":"In Git before
```

Data from the Lacework API

;

11.6.4. 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:

- Configure your app with `AuditLog.Read.All` API permissions, which Audit Logs API calls require.
- Notice that the API structures Azure Active Directory (AD) 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 the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Manage Sources** page's tiles or left nav. Click **+ Add New** 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 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. E.g., `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 Azure AD 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 Azure AD 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=${password}&scope=https://graph.microsoft.com/.default&client_id=${username}&grant_type=client_credentials` ``. When the Collector sends the request, it will populate the template with the **Username** and **Password** that you entered above.

- The **Token attribute**: The field that contains the Bearer token, within the Microsoft identity platform's response to the token request. Use `access_token`.
- The **Authorize expression**, 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

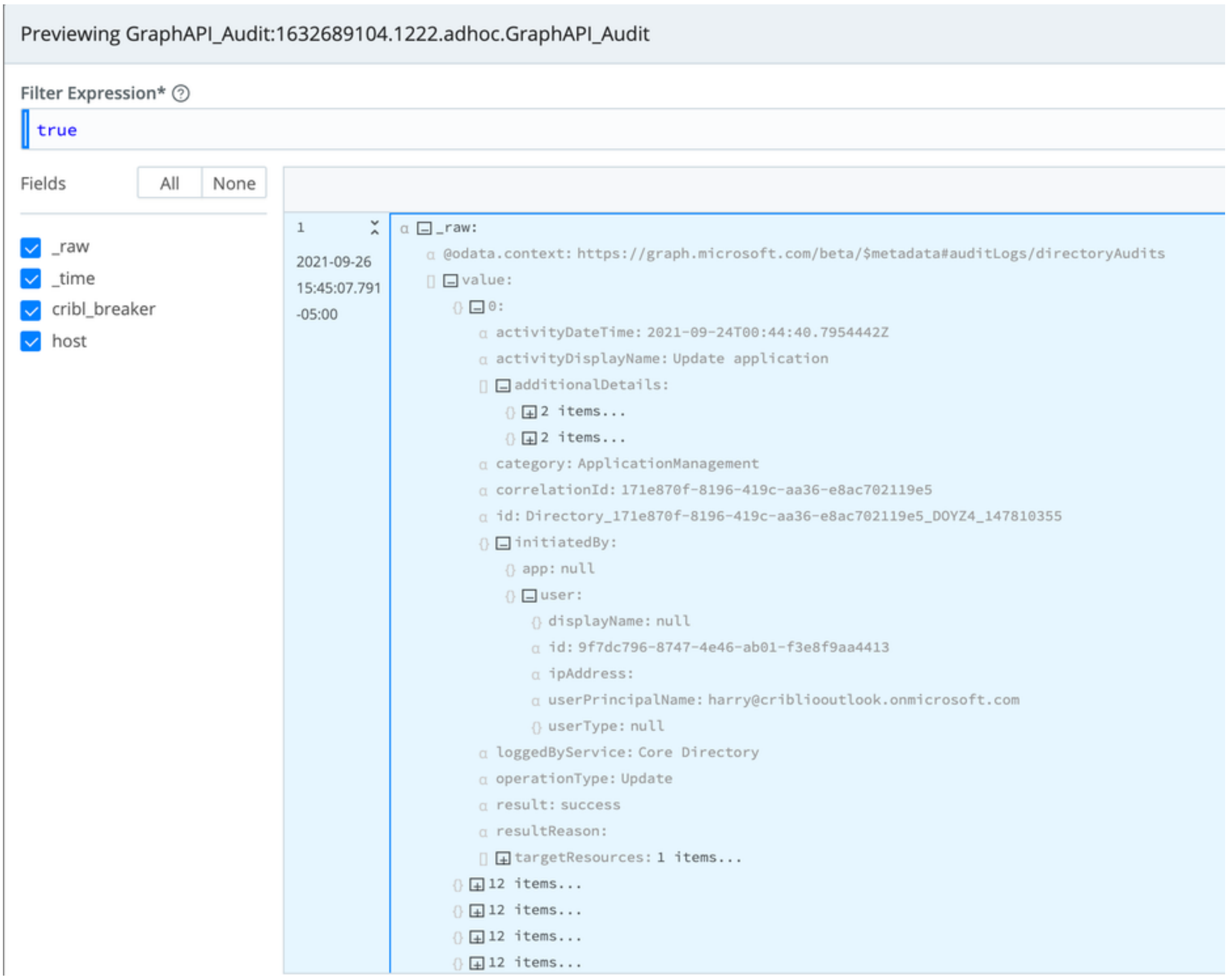
As explained above, the Microsoft Graph API delivers Azure Active Directory Audit Logs data as a single JSON object, with events nested under the `value` field. Cribl recommends using an Event Breaker to split this into separate name/value pairs.

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*

```
true
```

Fields

- ☑ _raw
- ☑ _time
- ☑ cribl_breaker
- ☑ host

```
1           _raw:
2021-09-26    @odata.context: https://graph.microsoft.com/beta/$metadata#auditLogs/directoryAudits
15:45:07.791  value:
-05:00          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

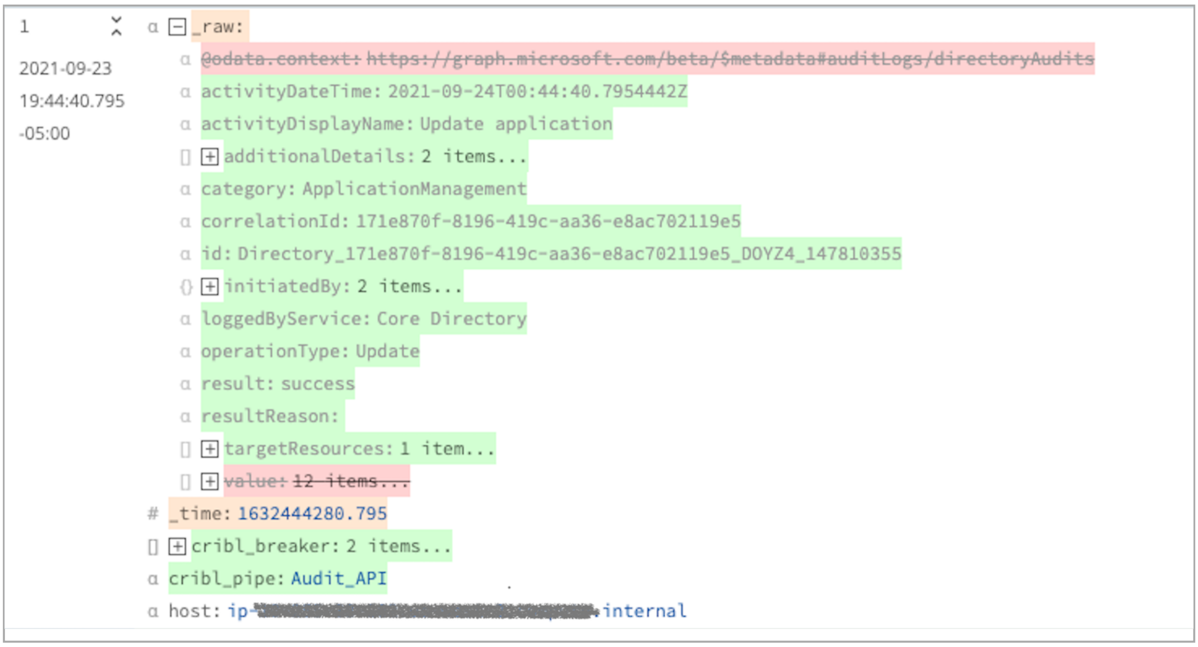
From the top menu, open **Knowledge** > **Event Breaker Rules**, then click **+ Add New** to open the **New Ruleset** modal. Enter an ID and a Description for the ruleset, then click **Rules** > **+ Add Rule** to open the **Rules** modal.

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:

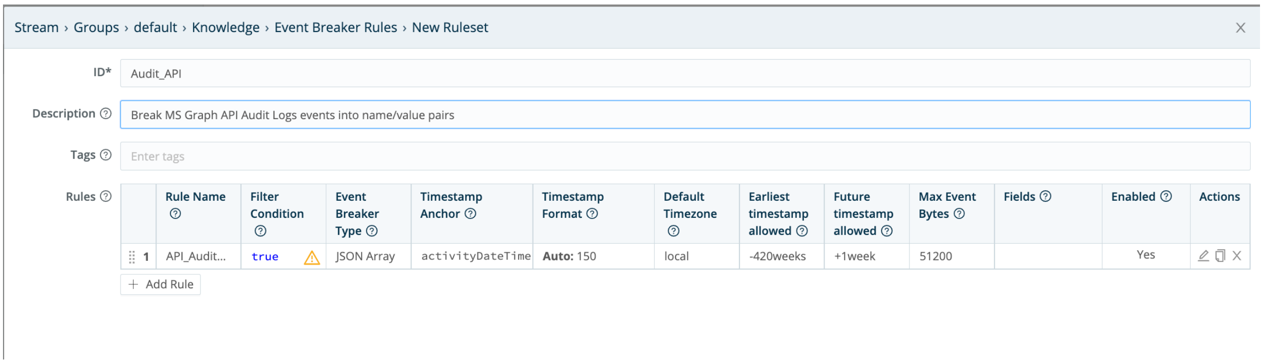
- From the **Event Breaker Type** drop-down, choose **JSON Array**. This detects that the `value` element is a JSON array, and breaks it into key-value pairs. It also discards the `@odata.context` element, because that falls outside the detected JSON array.
- In the **Timestamp Anchor** field, enter `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 the screenshot below. Click **Save**.



Stream › Groups › default › Knowledge › Event Breaker Rules › New Ruleset

ID*: Audit_API

Description: Break MS Graph API Audit Logs events into name/value pairs

Tags: Enter tags

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	API_Audit...	true ⚠	JSON Array	activityDateTime	**Auto:** 150	local	-420weeks	+1week	51200		Yes	

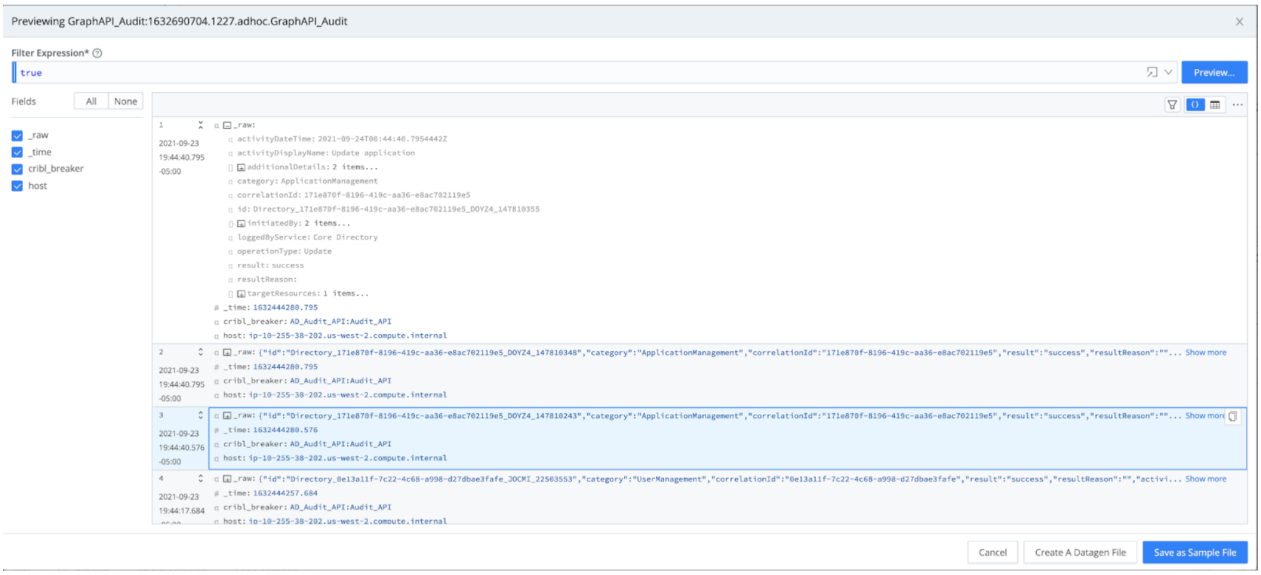
Creating an Event Breaker ruleset for Audit Logs data

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.



Filter Expression*: `true`

Fields: ☑ _raw ☑ _time ☑ cribl_breaker ☑ host

```
1                     _raw:
2021-09-23              activityDateTime: 2021-09-24T00:44:40.7954442Z
19:44:40.795            activityDisplayName: Update application
-05:00                  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                     _raw: {"id":"Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810348","category":"ApplicationManagement","correlationId":"171e870f-8196-419c-aa36-e8ac702119e5","result":"success","resultReason":""... Show more
2021-09-23            # _time: 1632444280.795
19:44:40.795          cribl_breaker: AD_Audit_API:Audit_API
-05:00                host: ip-10-255-38-202.us-west-2.compute.internal
3                     _raw: {"id":"Directory_171e870f-8196-419c-aa36-e8ac702119e5_DOYZ4_147810243","category":"ApplicationManagement","correlationId":"171e870f-8196-419c-aa36-e8ac702119e5","result":"success","resultReason":""... Show more
2021-09-23            # _time: 1632444280.576
19:44:40.576          cribl_breaker: AD_Audit_API:Audit_API
-05:00                host: ip-10-255-38-202.us-west-2.compute.internal
4                     _raw: {"id":"Directory_0e13a11f-7c22-4c68-a998-d27dbae3fafe_JOCMI_22503553","category":"UserManagement","correlationId":"0e13a11f-7c22-4c68-a998-d27dbae3fafe","result":"success","resultReason":"","activi... Show more
2021-09-23            # _time: 1632444257.684
19:44:17.684          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

;

11.6.5. 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`

NAME	VALUE EXPRESSION
`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 the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Collectors** > **REST** from the **Manage Sources** page's tiles or left nav. Click **+ Add New** 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.uri(`${sys_id}`)
```

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** buttons 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.

;

11.6.6. 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

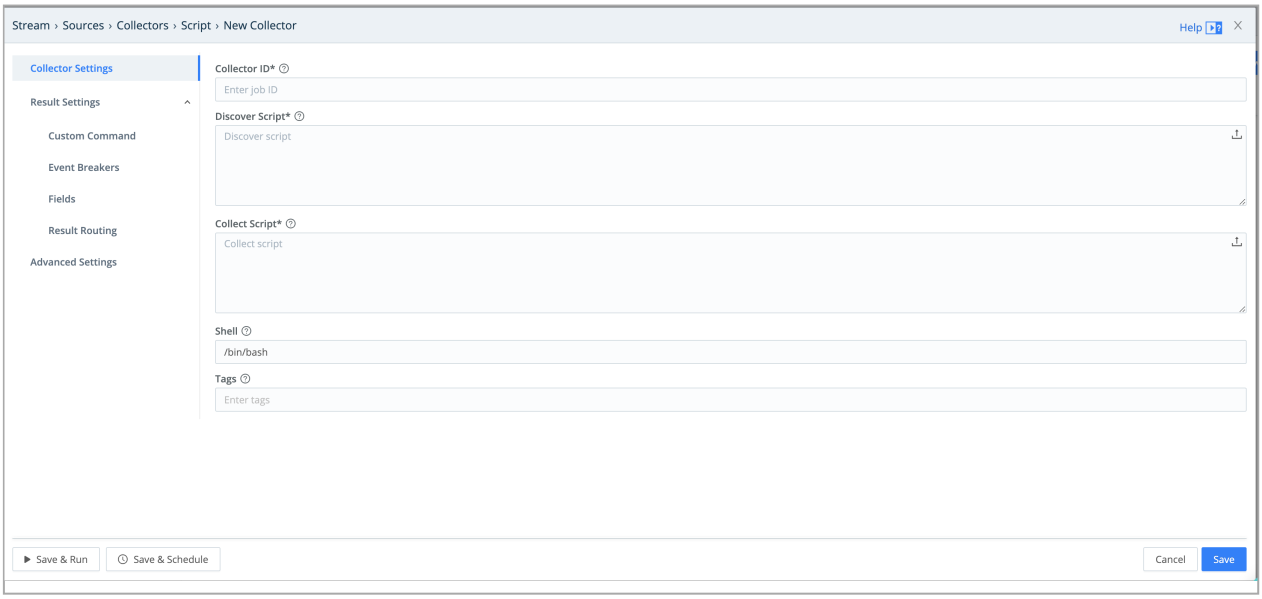
Each item, from the Discover phase's returned list of items, will trigger one Collect task for Cribl Stream to create and run. More on Collect:

- 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:



Shell
/bin/bash

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 per line in stdout."
    },
    "collectScript": {
      "type": "string",
      "title": "Collect Script",
      "minLength": 1,
      "description": "Script to run to perform data collections. Task passed in as $CRIBL_COLLECT_ARG. Should output results to stdout."
    },
    "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 }
= C.internal.HttpUtils;

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
Francisco"})
}

// 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
STATUS         PORTS                        NAMES
544370698fb5   cribl/cribl:3.4.1-RC1    "/sbin/entrypoint.sh…"   4 minutes ago
Up 4 minutes   0.0.0.0:19000->9000/tcp    nervous_wozniak
```

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';
```

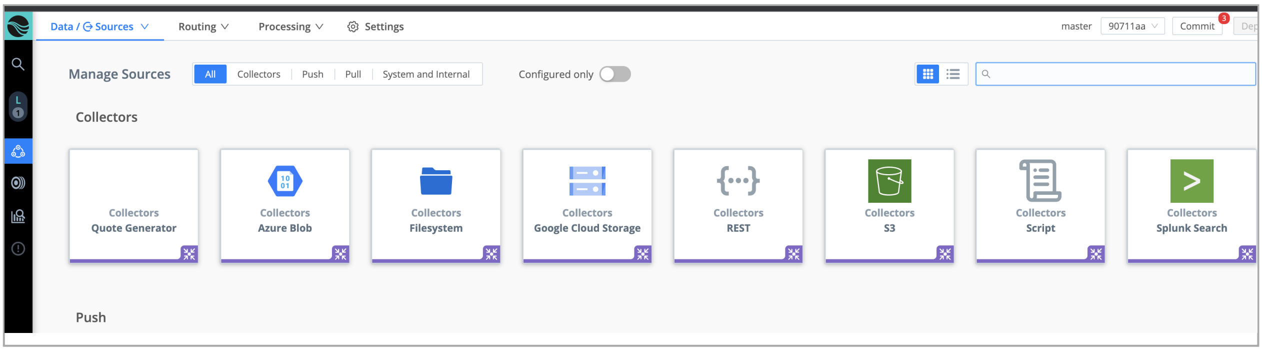
9. Next, restart Cribl Stream with the following command:

```
$ /opt/cribl/bin/cribl restart
```

10. 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.

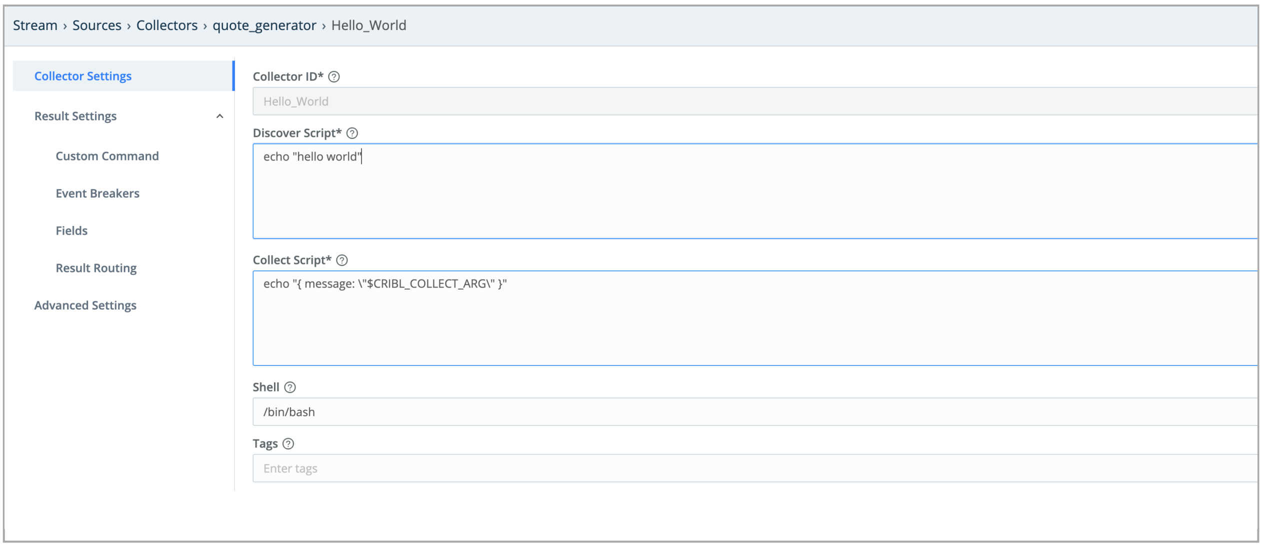


Quote Generator Collector (Missing an Icon)

2. Click **+ Add New** 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*:

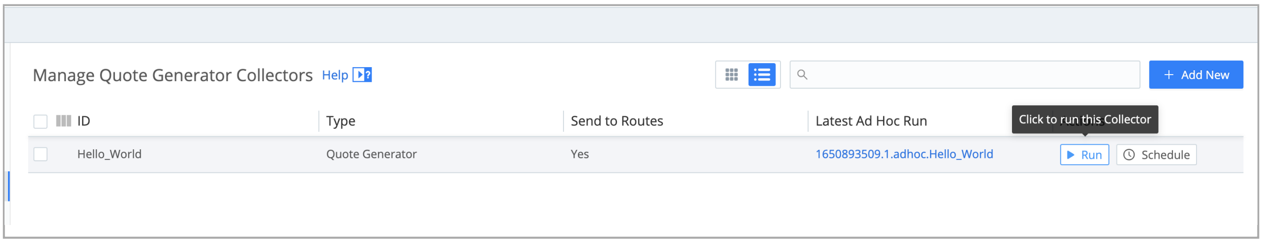
```
echo "{ message: \"$CRIBL_COLLECT_ARG\" }"
```

Shell: /bin/bash

Tags: Enter tags

Collector Settings Tab

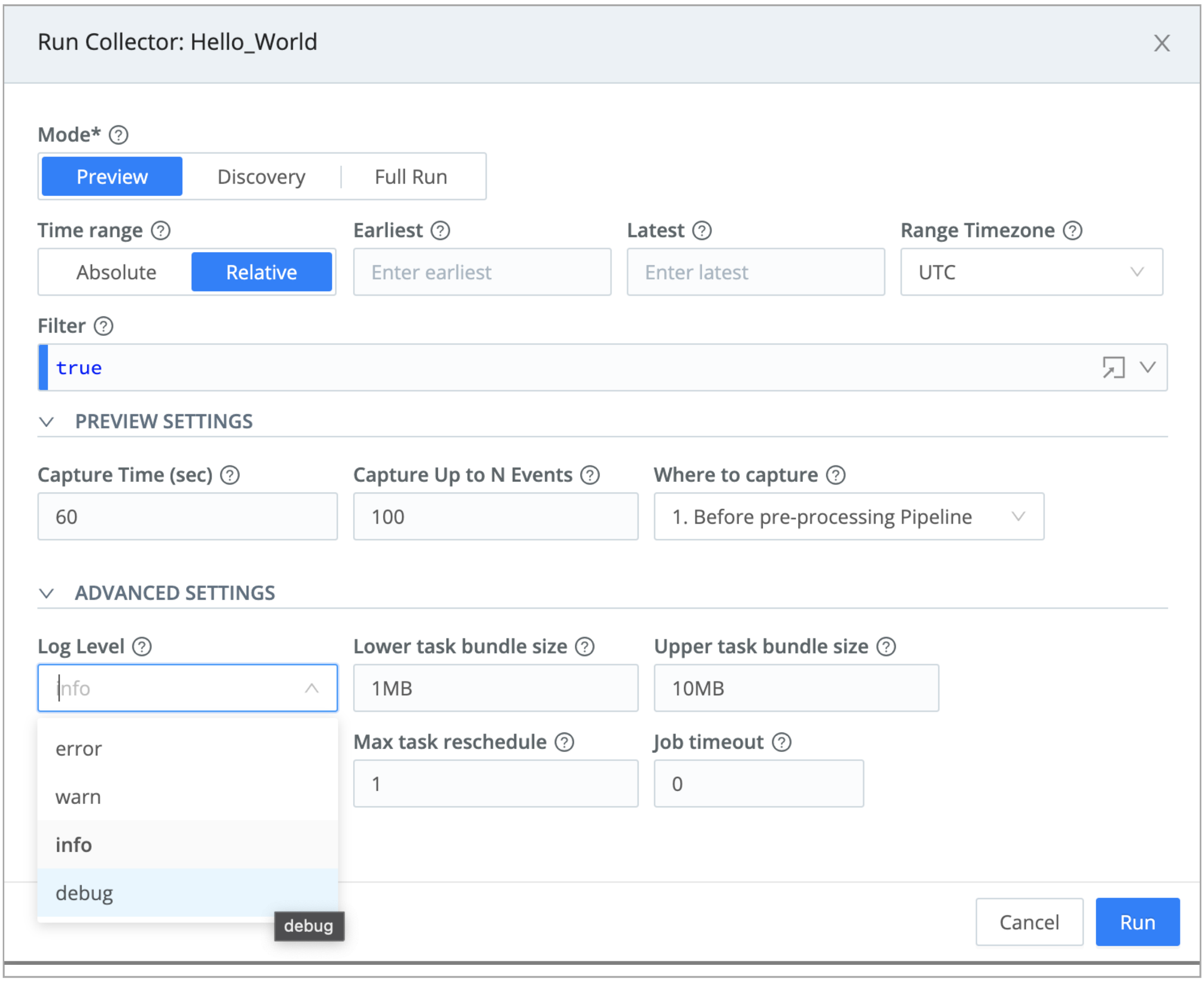
4. On the **Manage Quote Generator Collectors** page, click **Run** next to the Collector.


Manage Quote Generator Collectors
Help
+ Add New
Click to run this Collector
Run
Schedule

ID	Type	Send to Routes	Latest Ad Hoc Run
Hello_World	Quote Generator	Yes	1650893509.1.adhoc.Hello_World

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.



Run Collector: Hello_World

Mode*: Preview | Discovery | Full Run (Preview selected)

Time range: Absolute | Relative (Relative selected)

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: info

Lower task bundle size: 1MB

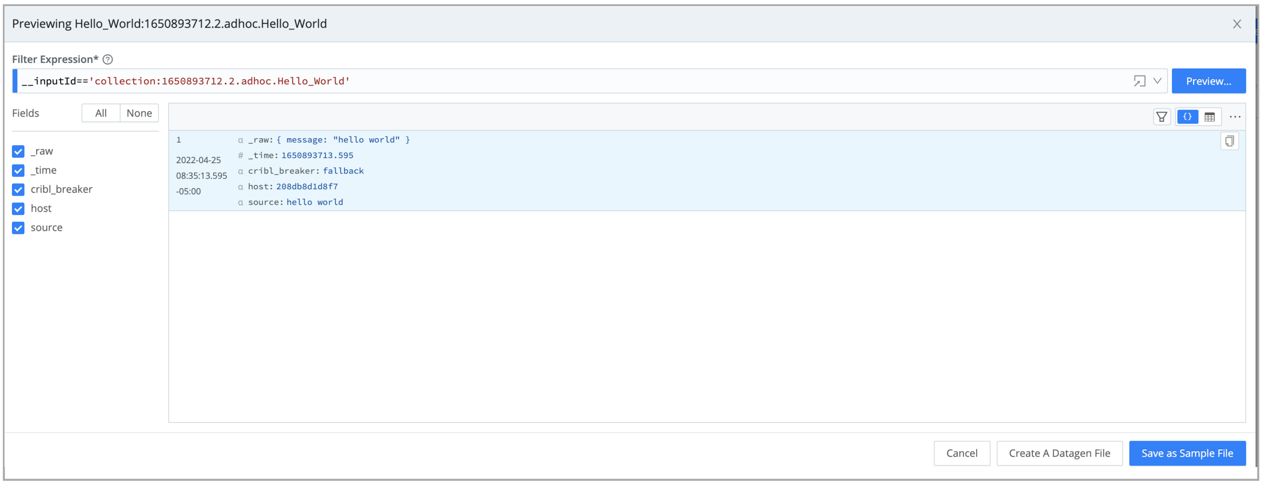
Upper task bundle size: 10MB

Max task reschedule: 1

Job timeout: 0

Run the Collector

6. After the Collector runs, the following event will be generated:



Previewing Hello_World:1650893712.2.adhoc.Hello_World

Filter Expression*: `__inputId=='collection:1650893712.2.adhoc.Hello_World'`

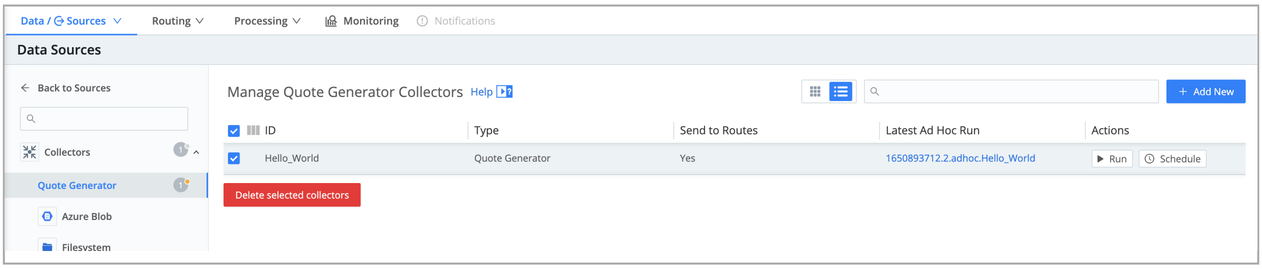
Fields: ☑ _raw ☑ _time ☑ cribl_breaker ☑ host ☑ source

```
1               α _raw: { message: "hello world" }
2022-04-25      # _time: 1650893713.595
08:35:13.595    α cribl_breaker: fallback
-05:00          α host: 208db8d1d8f7
                α 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	1650893712.2.adhoc.Hello_World	▶ Run · Schedule

Delete the Collector

Edit the Schema Files

1. Replace the contents of `conf.schema.json` with the following:

```
    {
      "type": "object",
      "title": "",
      "properties": {
        "autoGenerateNames": {
          "type": "boolean",
          "title": "Auto generate names",
          "description": "Turn on to autogenerate names, use num names to specify how many names to generate. Turn off to specify a static list of names",
          "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 will turn into a collection task."
                }
              }
            },
            {
              "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,
search worker logs for channel 'myCollector'

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
returning
  // 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
```

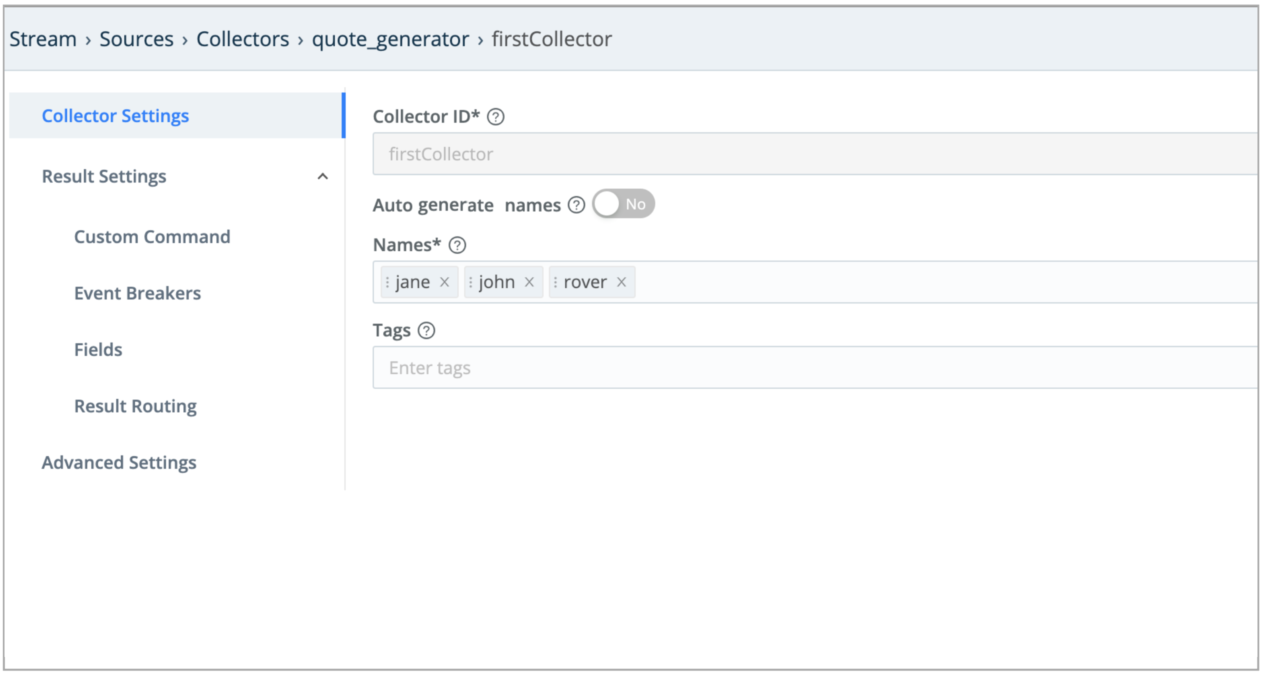
5. After Cribl Stream restarts, you must log out and log back in for the new Collector tile to display under Sources.

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`



Stream › Sources › Collectors › quote_generator › firstCollector

Collector ID*: firstCollector

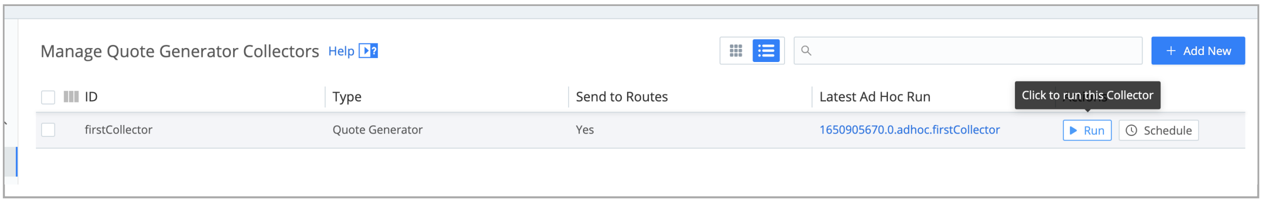
Auto generate names: No

Names*: jane, john, rover

Tags: Enter tags

New Collector Settings

3. On the **Manage Quote Generator Collectors** page click **Run** next to the Collector.



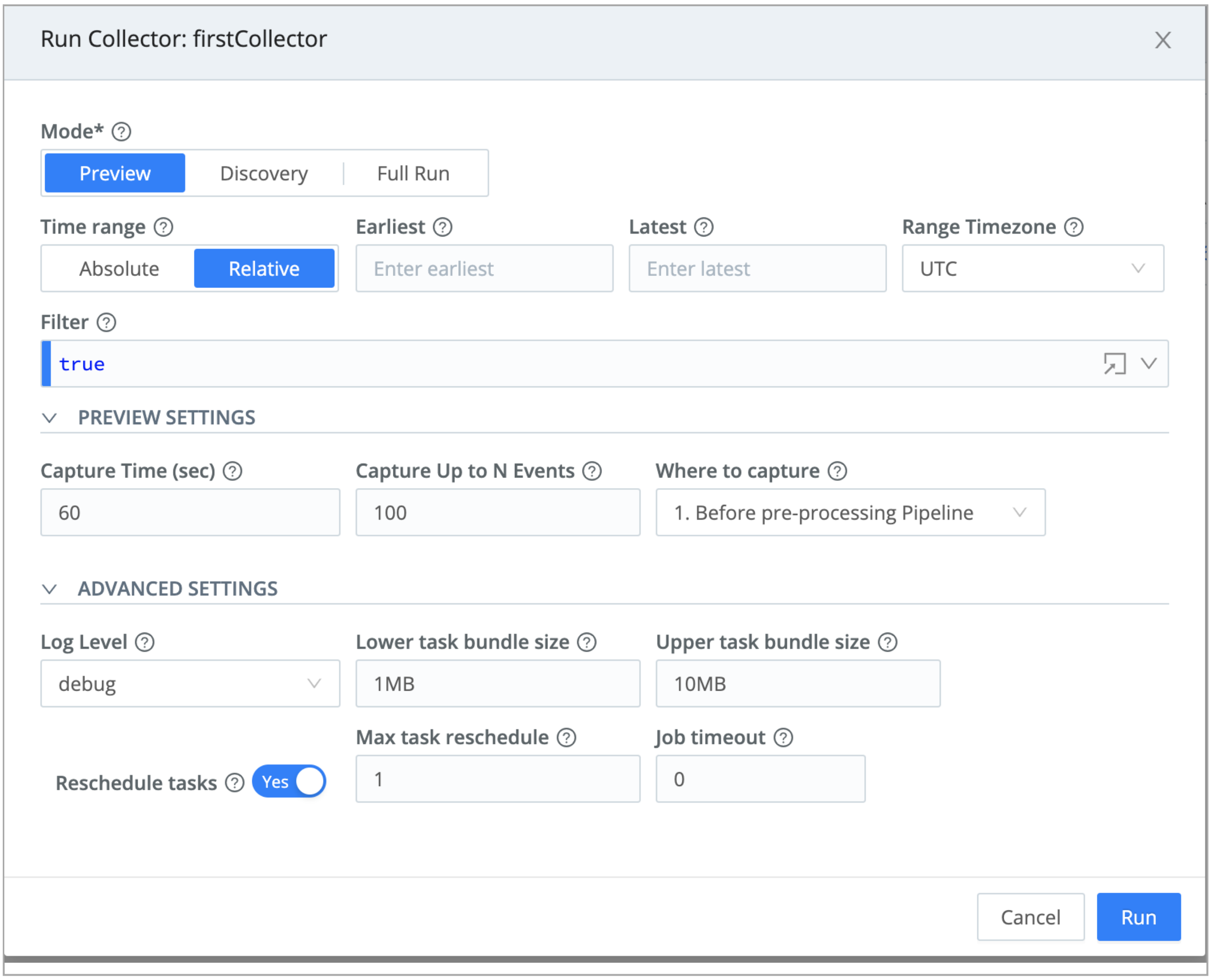
Manage Quote Generator Collectors

ID	Type	Send to Routes	Latest Ad Hoc Run	
firstCollector	Quote Generator	Yes	1650905670.0.adhoc.firstCollector	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

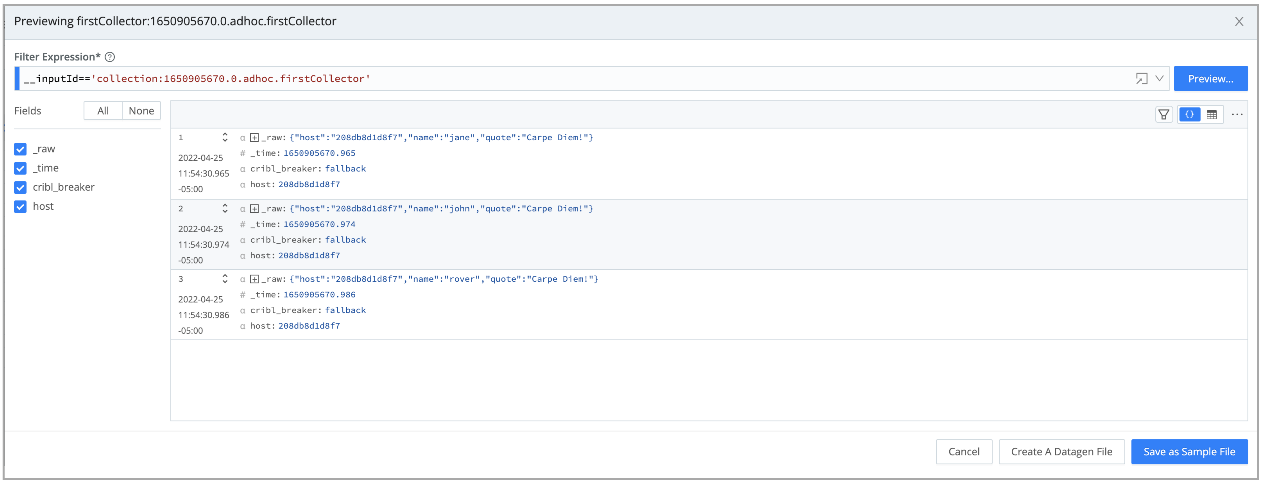
Max task reschedule: 1

Job timeout: 0

Reschedule tasks: Yes

Run Collector Settings

4. 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

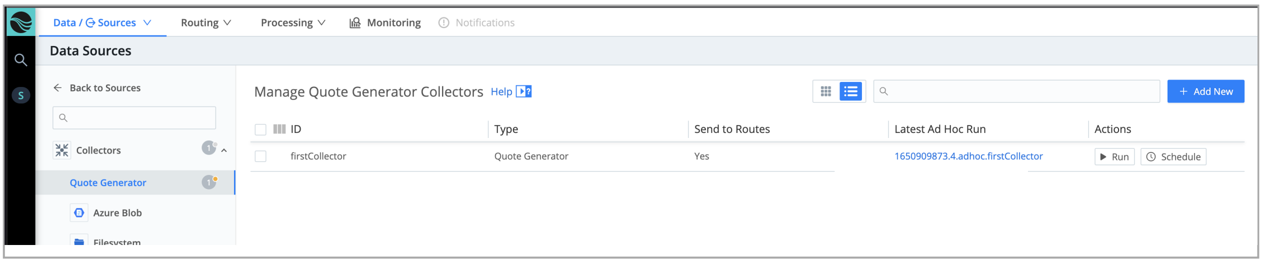
Filter Expression*: `__inputId=='collection:1650905670.0.adhoc.firstCollector'`

☑ _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
```

5. Close the **Preview** dialog.

6. On the **Manage Quote Generator Collectors** page, click **Latest ad hoc run**.

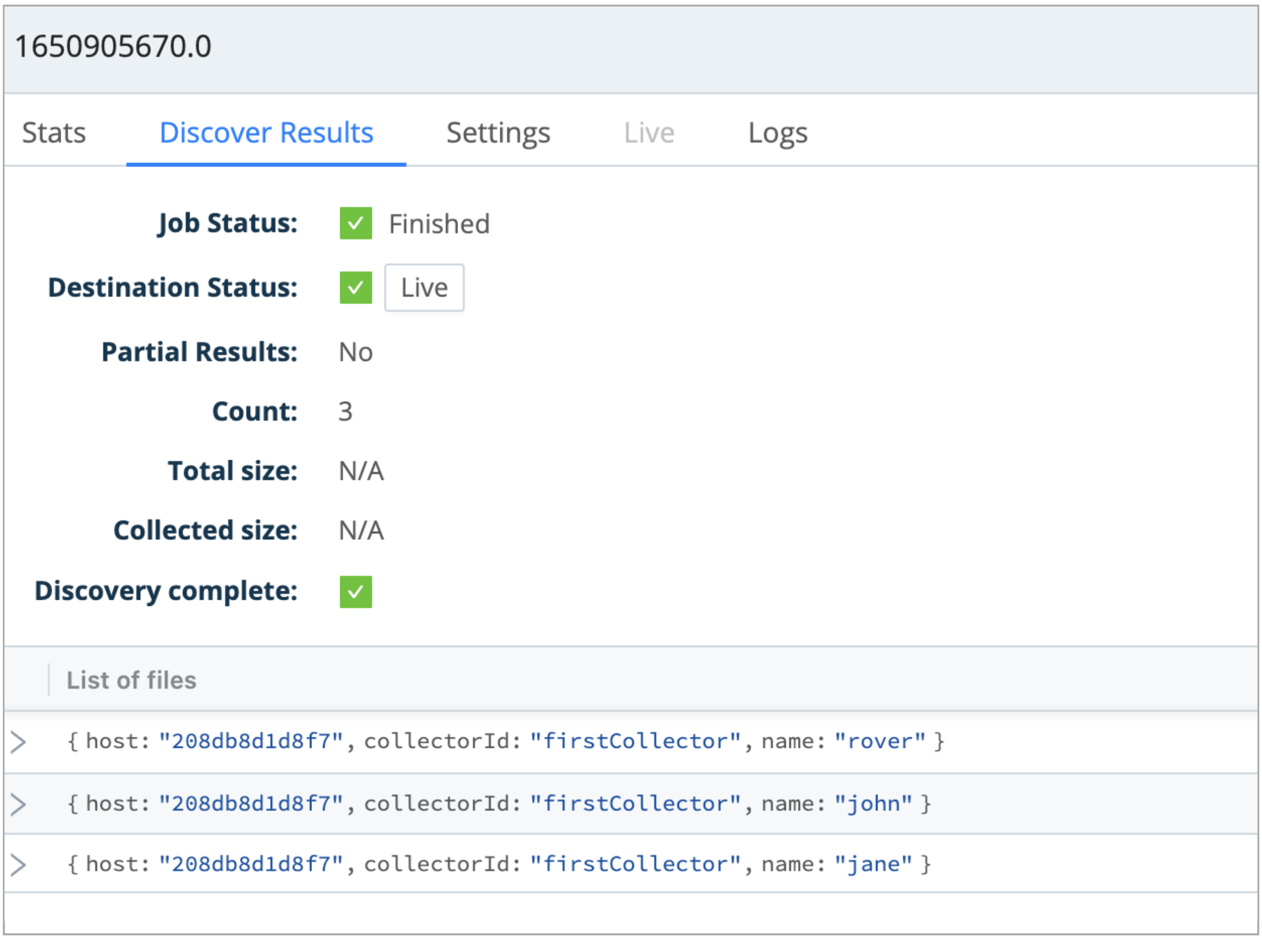


Manage Quote Generator Collectors

ID	Type	Send to Routes	Latest Ad Hoc Run	Actions
firstCollector	Quote Generator	Yes	1650909873.4.adhoc.firstCollector	▶ Run Schedule

Latest Ad Hoc Run column

7. 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

Job Status: ✓ Finished

Destination Status: ✓ Live

Partial Results: No

Count: 3

Total size: N/A

Collected size: N/A

Discovery complete: ✓

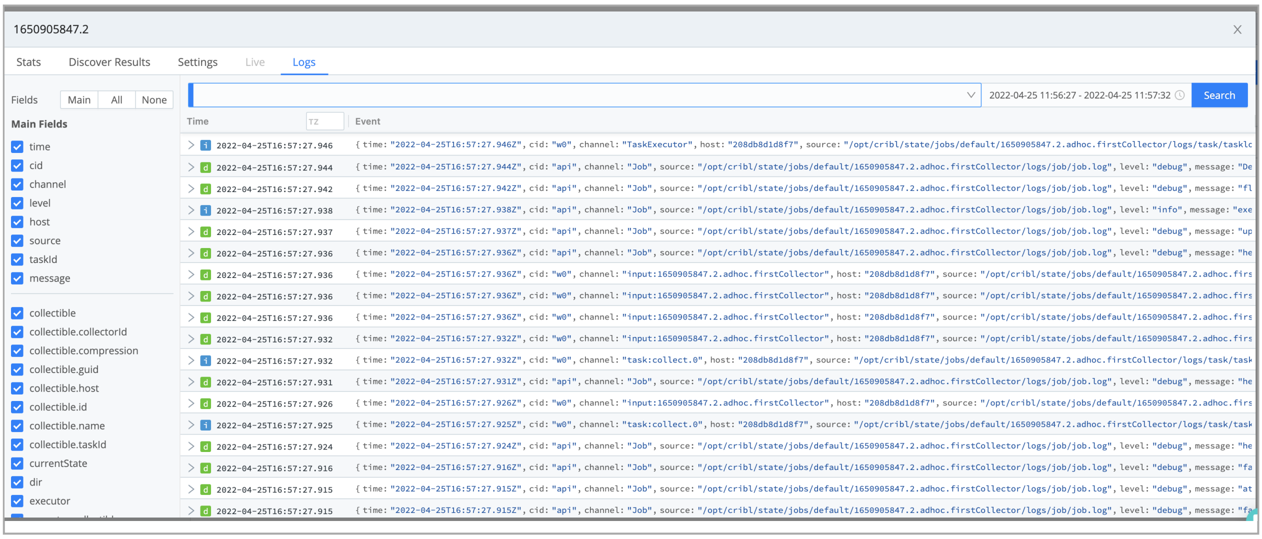
List of files

```
> { 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.

8. Click the **Logs** tab to view logs from the run. This is where you can locate anything logged by `job.logger`.



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

```
> 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`.

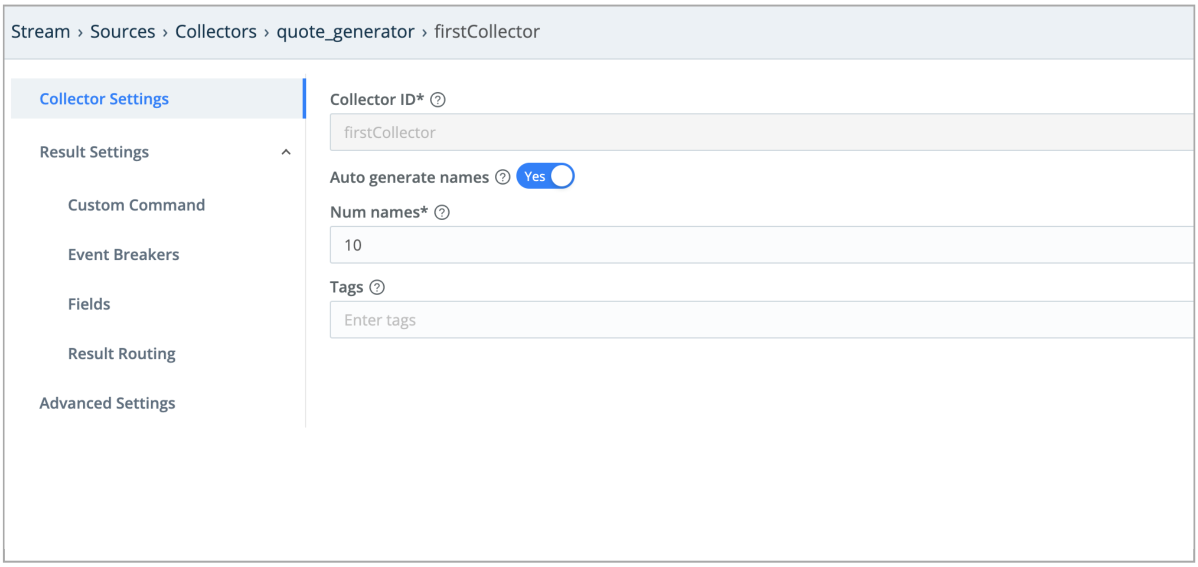
3. Next, update the Collector's `index.js` with the following content, to use this new package to auto-generate names:

```
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,
search worker logs for channel 'myCollector'

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:
names.length });
  }
  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 return a readable stream from the collect method. Here we are
returning
  // 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. Now, in Cribl Stream's UI, update the existing Collector to use the auto-generate feature. Add these settings, then click **Save**:

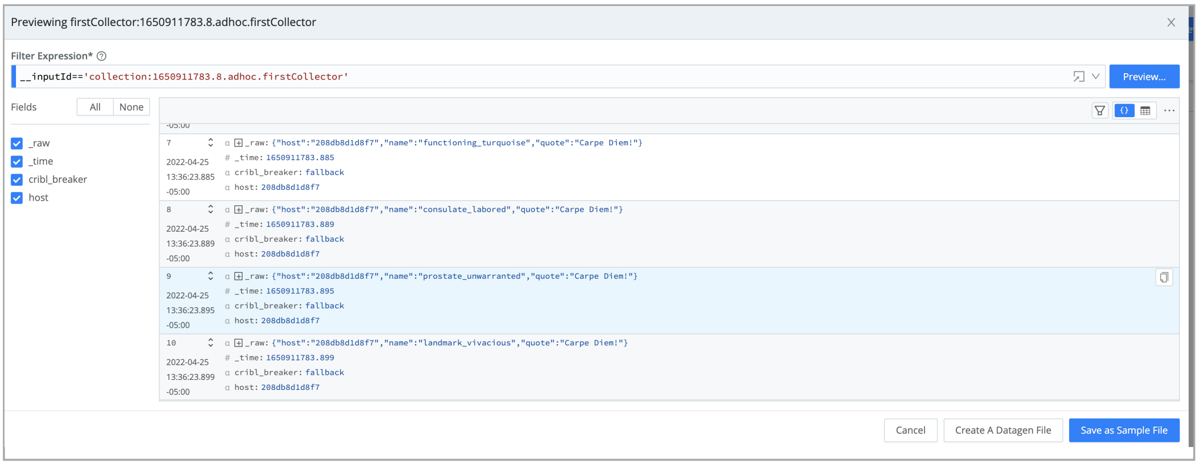
   - **Auto generate names**: `Yes`
   - **Num names:** `10`



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'`

☑ _raw
☑ _time
☑ cribl_breaker
☑ host

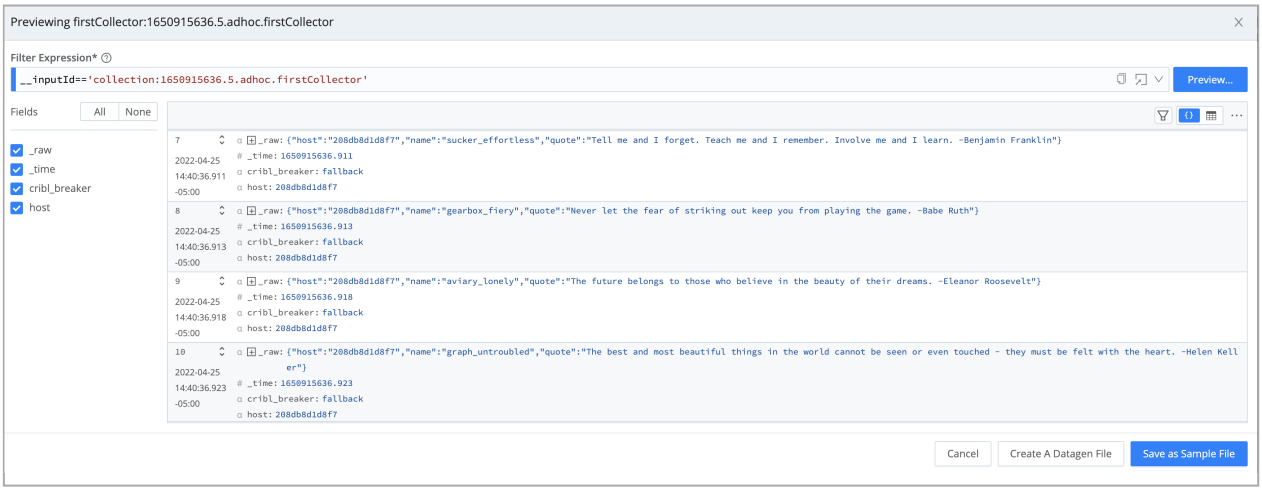
```
-05:00
7     _raw: {"host":"208db8d1d8f7","name":"functioning_turquoise","quote":"Carpe Diem!"}
2022-04-25   _time: 1650911783.885
13:36:23.885 cribl_breaker: fallback
-05:00       host: 208db8d1d8f7
8     _raw: {"host":"208db8d1d8f7","name":"consulate_labored","quote":"Carpe Diem!"}
2022-04-25   _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,
search worker logs for channel 'myCollector'

// Quotes to randomize
const quotes = [
  "Spread love everywhere you go. Let no one ever come to you without leaving
happier. -Mother Teresa",
  "When you reach the end of your rope, tie a knot in it and hang on. -Franklin
D. Roosevelt",
  "Always remember that you are absolutely unique. Just like everyone else. -
Margaret Mead",
  "Don't judge each day by the harvest you reap but by the seeds that you plant.
-Robert Louis Stevenson",
  "The future belongs to those who believe in the beauty of their dreams. -
Eleanor Roosevelt",
  "Tell me and I forget. Teach me and I remember. Involve me and I learn. -
Benjamin Franklin",
  "The best and most beautiful things in the world cannot be seen or even
touched - they must be felt with the heart. -Helen Keller",
  "It is during our darkest moments that we must focus to see the light. -
Aristotle",
  "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
leave a trail. -Ralph Waldo Emerson",
  "You will face many defeats in life, but never let yourself be defeated. -Maya
Angelou",
  "The greatest glory in living lies not in never falling, but in rising every
time we fall. -Nelson Mandela",
  "In the end, it's not the years in your life that count. It's the life in your
years. -Abraham Lincoln",
  "Never let the fear of striking out keep you from playing the game. -Babe
Ruth",
  "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 to
success when they gave up. -Thomas A. Edison",
  "You have brains in your head. You have feet in your shoes. You can steer
yourself any direction you choose. -Dr. Seuss",
  "If life were predictable it would cease to be life and be without flavor. -
Eleanor Roosevelt",
  "In the end, it's not the years in your life that count. It's the life in your
years. -Abraham Lincoln",
  "Life is a succession of lessons which must be lived to be understood. -Ralph
Waldo Emerson",
  "You will face many defeats in life, but never let yourself be defeated. -Maya
Angelou",
]

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:
names.length });
  }
  for (let i = 0; i < names.length; i++) {
    await job.addResult({"name": names[i]});
  }
}

exports.collect = async (collectible, job) => {
  const quote = quotes[Math.trunc(Math.random()*100)%quotes.length];
  // Must return a readable stream from the collect method. Here we are
returning
  // the result string wrapped in a Readable.
  const result = { host, name: collectible.name, quote };
  job.logger().info('collect returning quote', { collectible, quote });
  const s = new Readable();
  s.push(JSON.stringify(result));
  s.push(null);

  // Return readable to stream collected data
  return Promise.resolve(s)
};
```

7. Run the Collector to randomly retrieve the quotes from the list:



```
__inputId=='collection:1650915636.5.adhoc.firstCollector'
```

```
7                    α ⊞ _raw: {"host":"208db8d1d8f7","name":"sucker_effortless","quote":"Tell me and I forget. Teach me and I remember. Involve me and I learn. -Benjamin Franklin"}
2022-04-25           # _time: 1650915636.911
14:40:36.911         α cribl_breaker: fallback
-05:00               α host: 208db8d1d8f7
8                    α ⊞ _raw: {"host":"208db8d1d8f7","name":"gearbox_fiery","quote":"Never let the fear of striking out keep you from playing the game. -Babe Ruth"}
2022-04-25           # _time: 1650915636.913
14:40:36.913         α cribl_breaker: fallback
-05:00               α host: 208db8d1d8f7
9                    α ⊞ _raw: {"host":"208db8d1d8f7","name":"aviary_lonely","quote":"The future belongs to those who believe in the beauty of their dreams. -Eleanor Roosevelt"}
2022-04-25           # _time: 1650915636.918
14:40:36.918         α cribl_breaker: fallback
-05:00               α host: 208db8d1d8f7
10                   α ⊞ _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 Kell
                              er"}
2022-04-25           # _time: 1650915636.923
14:40:36.923         α cribl_breaker: fallback
-05:00               α 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
STATUS         PORTS                         NAMES
208db8d1d8f7   cribl/cribl:latest   "/sbin/entrypoint.sh…"   6 hours ago    Up 6
hours    0.0.0.0:19000->9000/tcp    serene_galileo
```

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.

;

11.7. Securing Cribl Stream

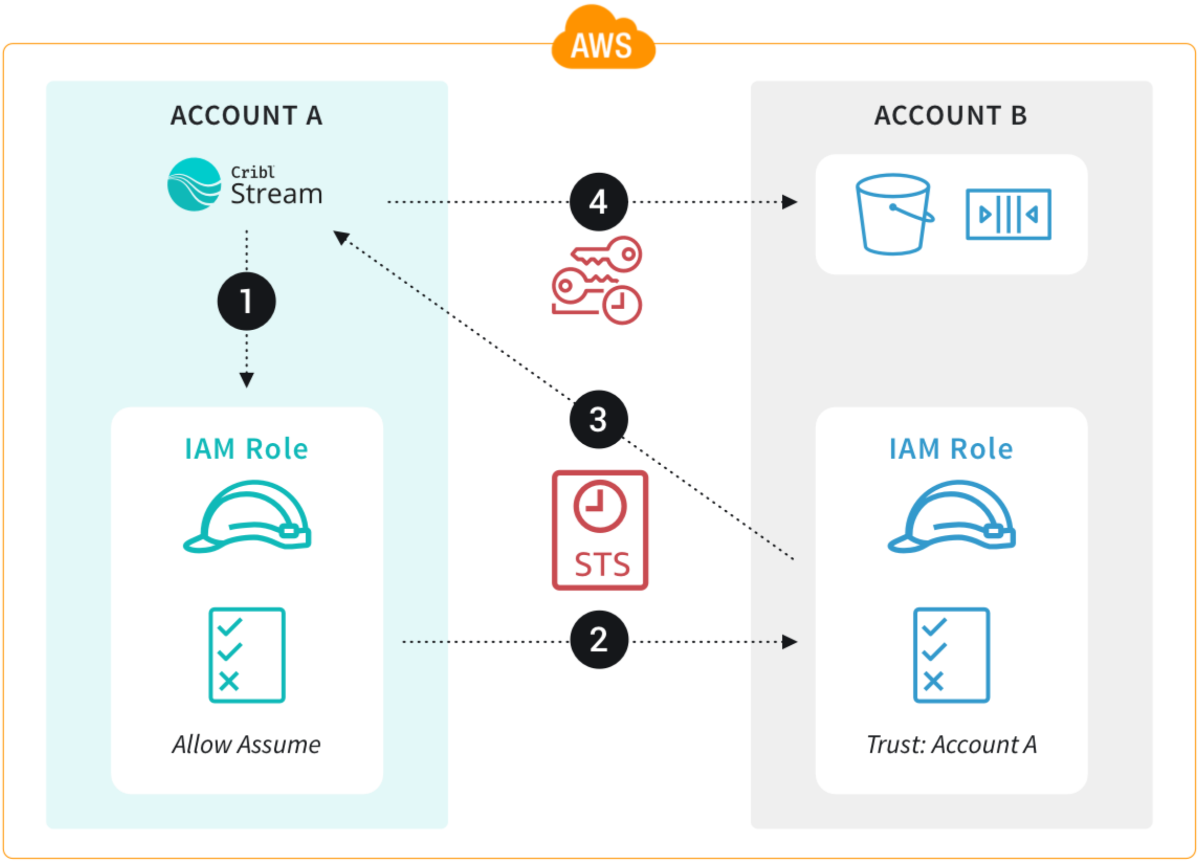
11.7.1. AWS Cross-Account Data Collection

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.

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:

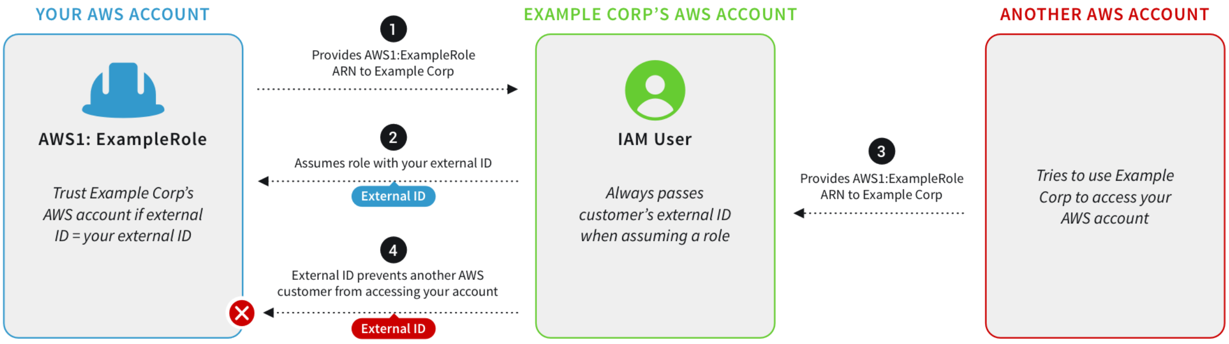
```
{
 "Version": "2012-10-17",
 "Statement": [
   {
     "Effect": "Allow",
     "Principal": {
       "AWS": "arn:aws:iam::111111111111:role/account-a-logstream-assumerole-role"
     },
     "Action": "sts:AssumeRole",
     "Condition": {
       "StringEquals": {
         "sts:ExternalId": "cribl-s3cre3t"
       }
     }
   }
 ]
}
```

> 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/worker-in.logstream"
     },
     "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-managedCribl Cloud Worker, skp 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:

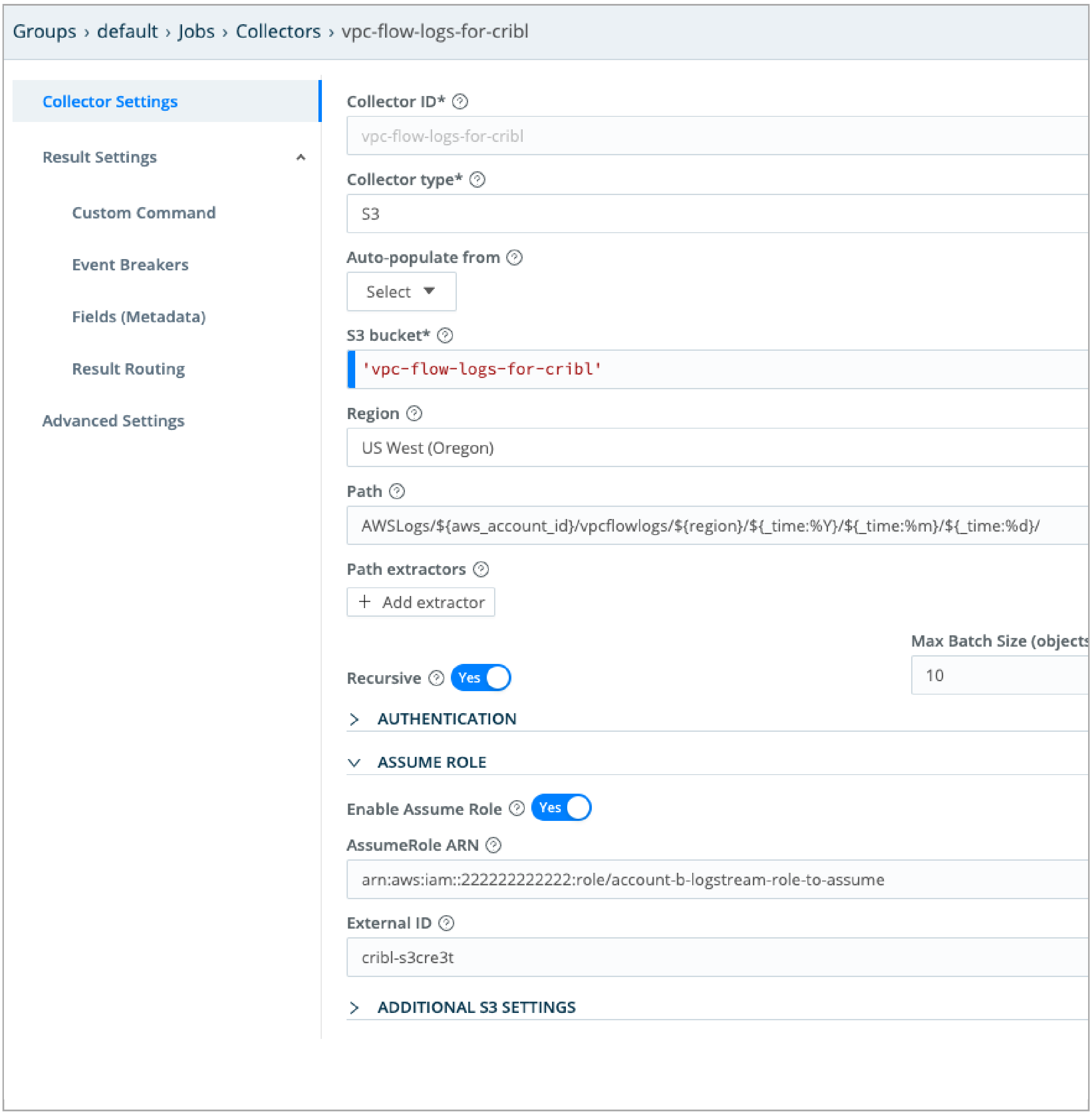
```
{
 "Version": "2012-10-17",
 "Statement": {
   "Effect": "Allow",
   "Action": "sts:AssumeRole",
   "Resource": "arn:aws:iam::222222222222:role/account-b-logstream-role-to-assume"
 }
}
```

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.



Groups › default › Jobs › Collectors › vpc-flow-logs-for-cribl

Collector ID*: vpc-flow-logs-for-cribl

Collector type*: S3

Auto-populate from: Select

S3 bucket*: 'vpc-flow-logs-for-cribl'

Region: US West (Oregon)

Path: AWSLogs/${aws_account_id}/vpcflowlogs/${region}/${_time:%Y}/${_time:%m}/${_time:%d}/

Path extractors: + Add extractor

Max Batch Size (objects: 10

Recursive: Yes

AUTHENTICATION

ASSUME ROLE

Enable Assume Role: Yes

AssumeRole ARN: arn:aws:iam::222222222222:role/account-b-logstream-role-to-assume

External ID: cribl-s3cre3t

ADDITIONAL S3 SETTINGS

Cribl Stream Assume Role configuration

;

11.7.2. 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.

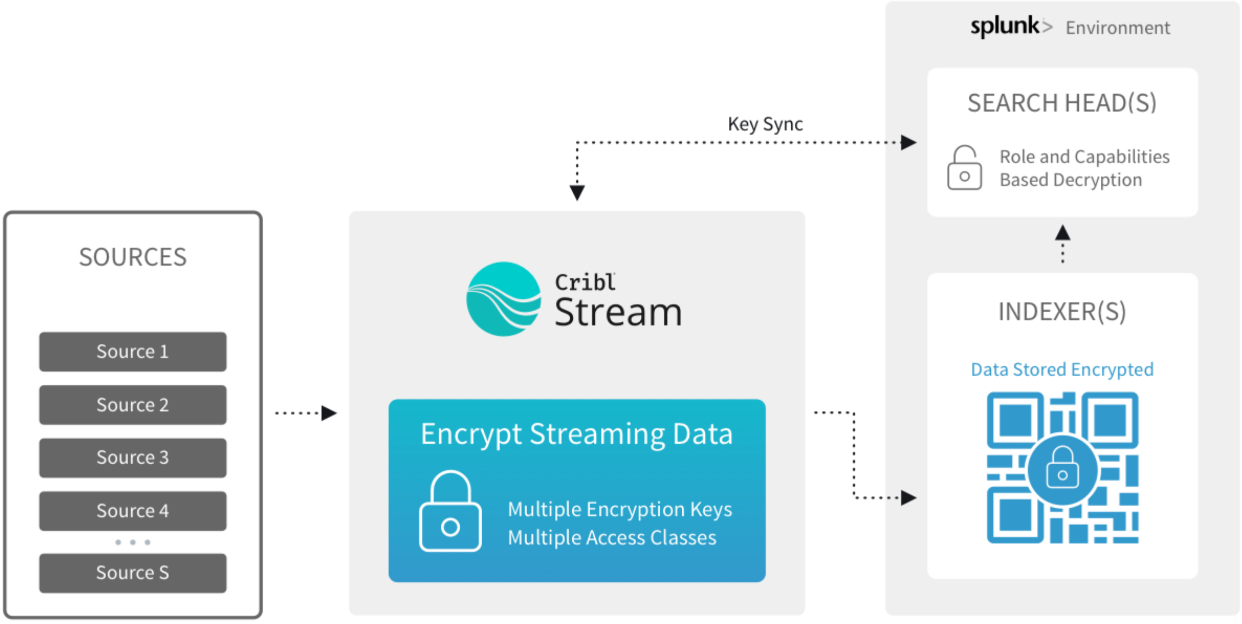
Encrypting in Cribl Stream and Decrypting in Splunk

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.

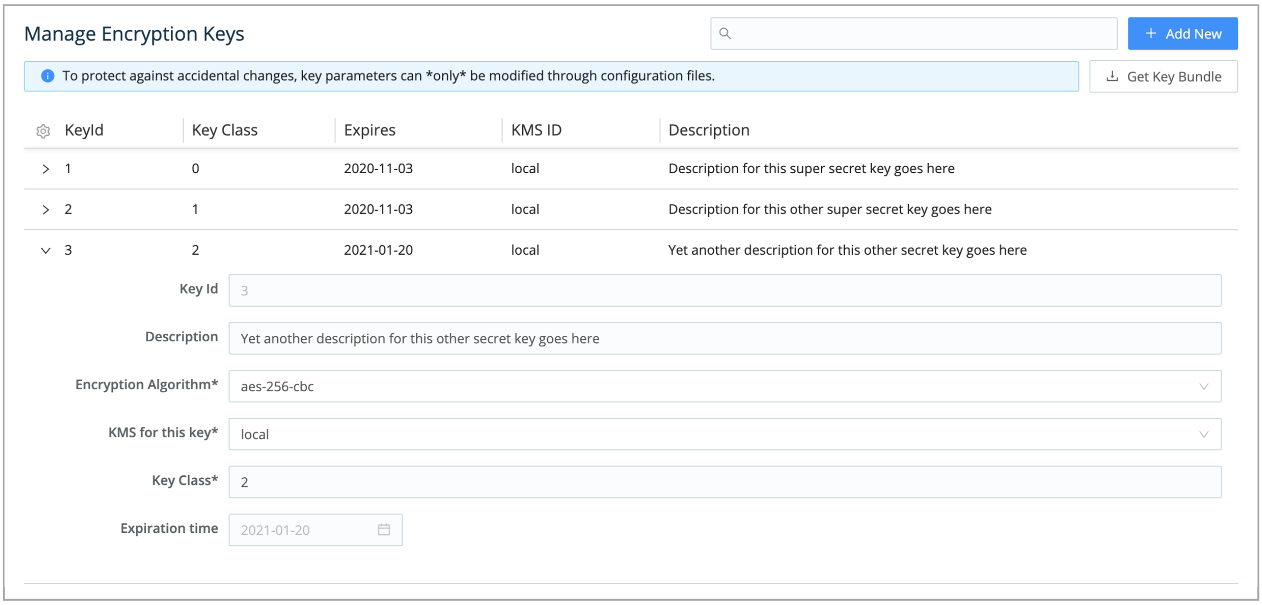


Encrypting in Cribl Stream, decrypting in Splunk

Example

Encryption Side

- Generate keys via the UI, in global ⚙ **Settings** (lower left) > **Security > Encryption Keys**:



Manage Encryption Keys

ⓘ To protect against accidental changes, key parameters can *only* be modified through configuration files.

KeyId	Key Class	Expires	KMS ID	Description
1	0	2020-11-03	local	Description for this super secret key goes here
2	1	2020-11-03	local	Description for this other super secret key goes here
3	2	2021-01-20	local	Yet another description for this other secret key goes here

Key Id	3
Description	Yet another description for this other secret key goes here
Encryption Algorithm*	aes-256-cbc
KMS for this key*	local
Key Class*	2
Expiration time	2021-01-20

Adding a new encryption key

- Or generate one or more keys via the CLI. 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`:

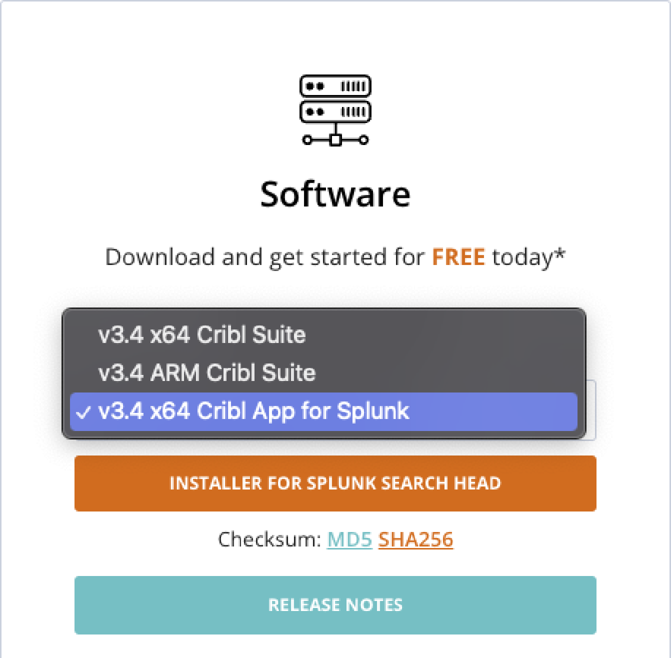
```
$CRIBL_HOME/bin/cribl keys add -c 1 -i -g uk ... $CRIBL_HOME/bin/cribl keys add -c
<N> -i -g uk
```

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`.

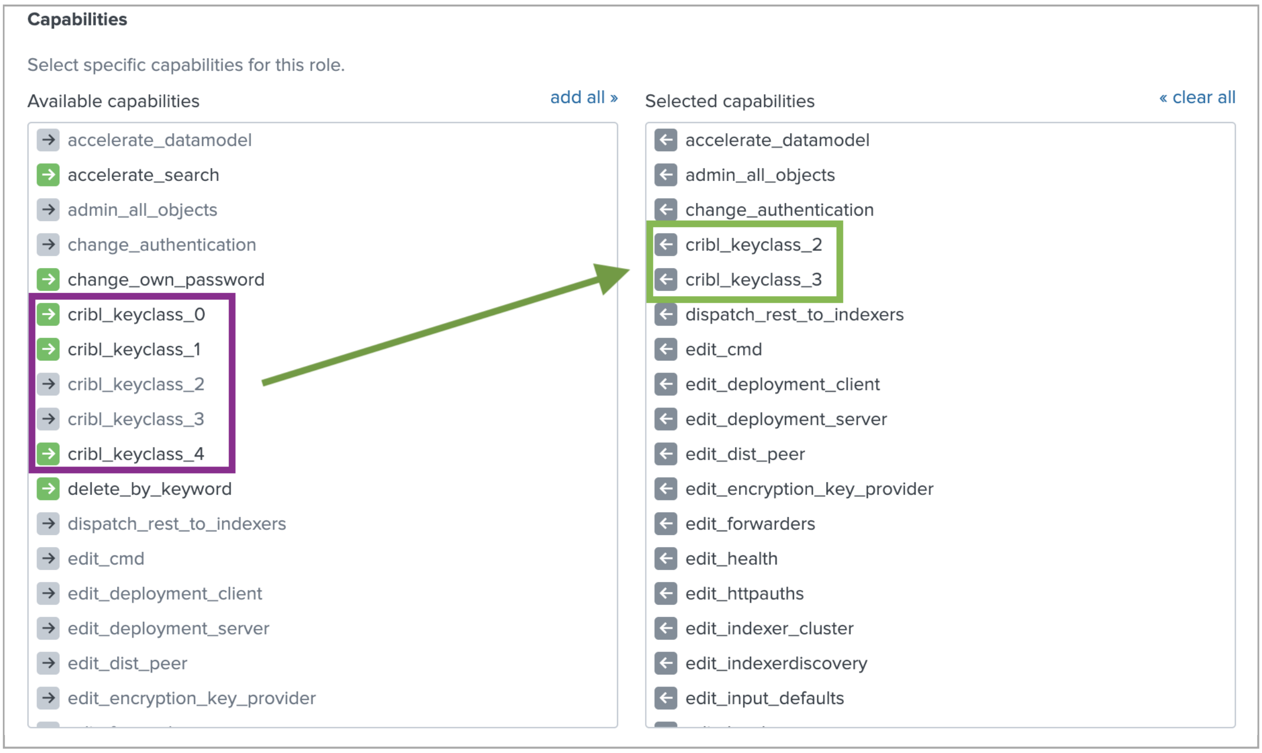

Software
Download and get started for FREE today*
v3.4 x64 Cribl Suite
v3.4 ARM Cribl Suite
✓ v3.4 x64 Cribl App for Splunk
INSTALLER FOR SPLUNK SEARCH HEAD
Checksum: MD5 SHA256
RELEASE NOTES

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.



Capabilities

Select specific capabilities for this role.

Available capabilities — add all »

- accelerate_datamodel
- accelerate_search
- admin_all_objects
- change_authentication
- change_own_password
- cribl_keyclass_0
- cribl_keyclass_1
- cribl_keyclass_2
- cribl_keyclass_3
- cribl_keyclass_4
- delete_by_keyword
- dispatch_rest_to_indexers
- edit_cmd
- edit_deployment_client
- edit_deployment_server
- edit_dist_peer
- edit_encryption_key_provider

Selected capabilities — « clear all

- accelerate_datamodel
- admin_all_objects
- change_authentication
- cribl_keyclass_2
- cribl_keyclass_3
- dispatch_rest_to_indexers
- edit_cmd
- edit_deployment_client
- edit_deployment_server
- edit_dist_peer
- edit_encryption_key_provider
- edit_forwarders
- edit_health
- edit_httpauths
- edit_indexer_cluster
- edit_indexerdiscovery
- edit_input_defaults

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 the **Get Key Bundle** button.

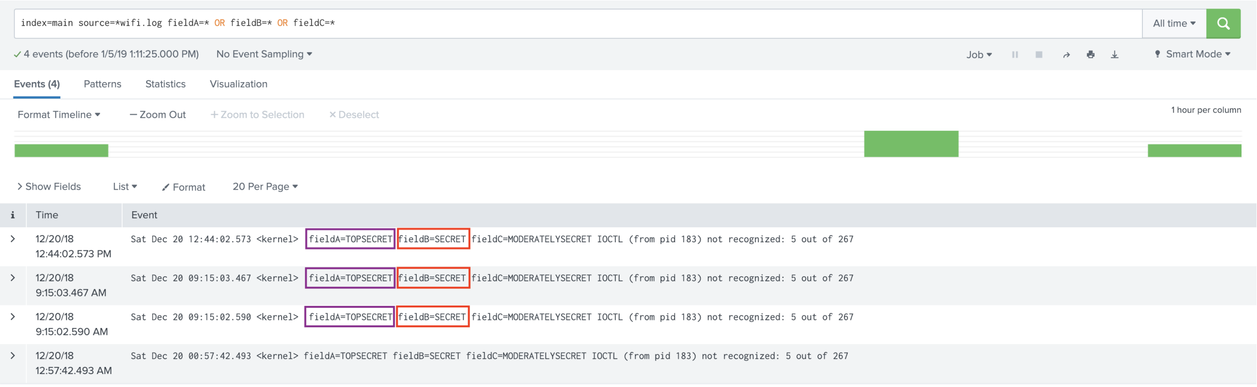
Sync/copy these files over to their counterparts on the search head (decryption side). In a non-Splunk integration, you would copy these assets to wherever decryption will take place.

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).

Usage

Before Encryption: Sample un-encrypted events. Notice the values of `fieldA` and `fieldB`.

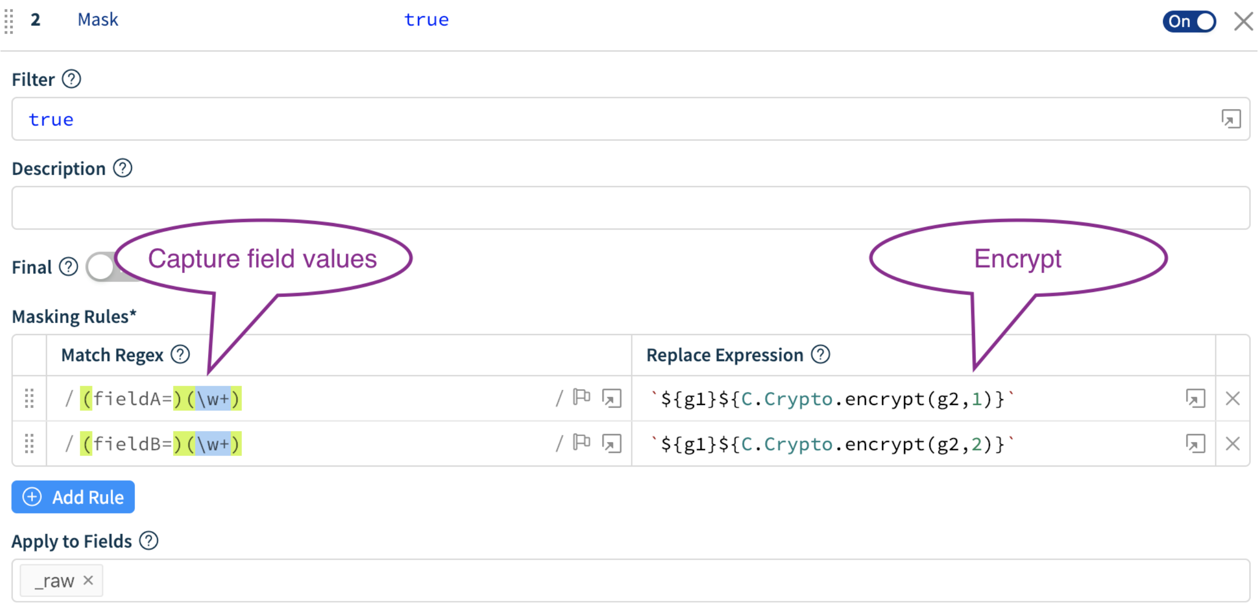


```
index=main source=*wifi.log fieldA=* OR fieldB=* OR fieldC=*
```

i	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*

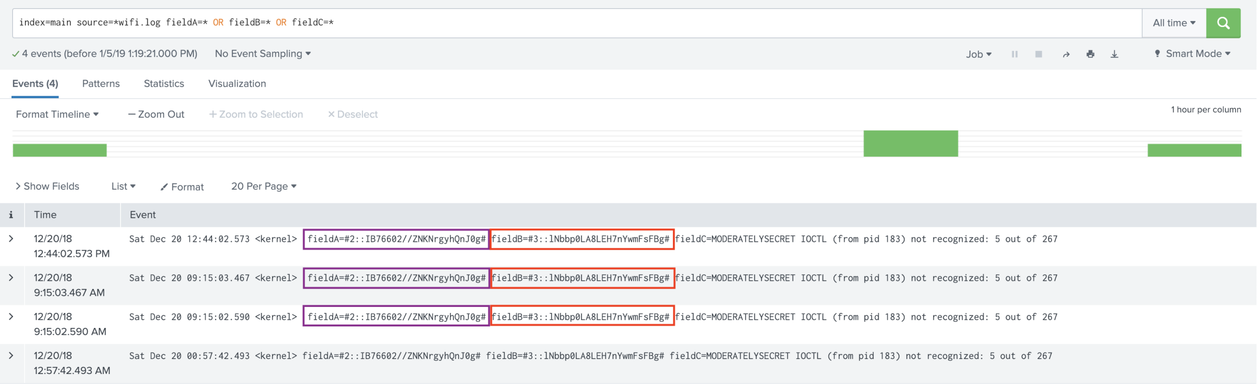
Match Regex	Replace Expression
`/(fieldA=)(\w+)/`	`` `${g1}${C.Crypto.encrypt(g2,1)}` ``
`/(fieldB=)(\w+)/`	`` `${g1}${C.Crypto.encrypt(g2,2)}` ``

Add Rule

Apply to Fields: _raw ×

Encrypting two fields with separate key classes

After Encryption: again, notice the values of `fieldA` and `fieldB`.

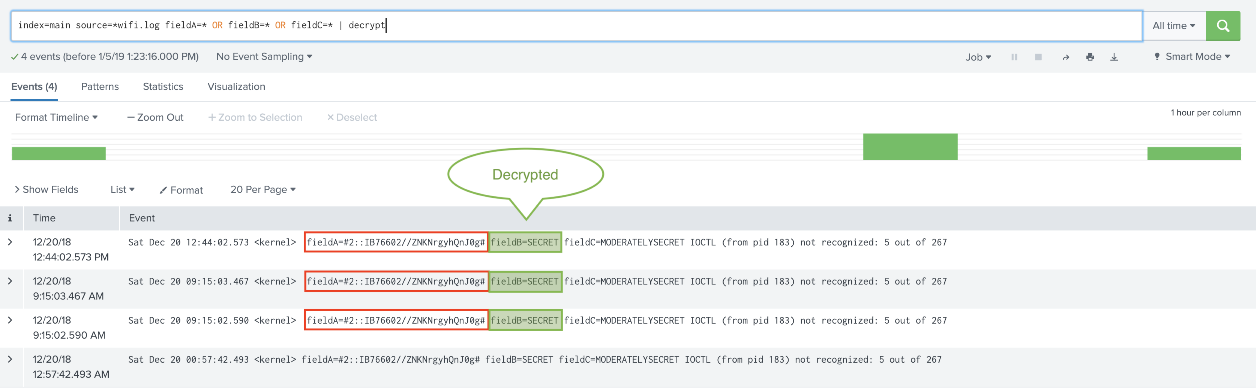


```
index=main source=*wifi.log fieldA=* OR fieldB=* OR fieldC=*
```

i	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
```

i	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

;

11.7.3. 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 denied

cribl.service: Control process exited, code=exited status=203
cribl.service: Failed to execute command: No such file or directory
cribl.service: Failed at step EXEC spawning /bin/rm -f /opt/cribl/pid/cribl.pid:
Permission denied
```

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 bin
drwxr-xr-x. 5 cribl cribl unconfined_u:object_r:user_home_t:s0  49 Feb 16 08:20 data
drwxr-xr-x. 6 cribl cribl unconfined_u:object_r:user_home_t:s0  59 Feb 16 08:31
default
drwxr-xr-x. 3 cribl cribl unconfined_u:object_r:user_home_t:s0  22 Feb 16 08:31
thirdparty
```

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 thirdparty
```

;

11.7.4. System Proxy Configuration

You can direct all outbound HTTP/S requests to go through proxy servers. You do so by setting a few environment variables before starting Cribl 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.

`HTTP_PROXY` examples:

```
$ export HTTP_PROXY=http://10.15.20.25:1234
$ export HTTP_PROXY=http://proxy.example.com:1234
```

`HTTPS_PROXY` examples:

```
$ export HTTPS_PROXY=http://10.15.20.25:5678
$ export HTTPS_PROXY=http://proxy.example.com:5678
```

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

> ### Restarts and Case Conflicts
>
> Initial configuration of, and changes to, these variables require restarting Cribl Stream on the affected Nodes, if the application is already running when you apply the changes.
>
> The environment variables' names can be either uppercase or lowercase. However, if you set duplicate versions of the same name, the lowercase version takes precedence. E.g., if you've set both `HTTPS_PROXY` and `https_proxy`, the IP address specified in `https_proxy` will take effect.

HTTP and/or HTTPS?

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

You might configure certain other Cribl 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:

```
[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>"
```

`NO_PROXY` Usage

- Cribl recommends including the Leader Node's host name in the `NO_PROXY` list.
- Within the list, separate the host/domain names with commas or spaces.
- Optionally, each host/domain entry can be followed by a port. If specified, the port must match. If not specified, the protocol's default port is assumed.
- If specified, subdomain names must match. E.g., `NO_PROXY=foo.example.com` will send requests directly to https://foo.example.com, but https://bar.example.com requests will go through the proxy.
- You can use leading wildcards like `NO_PROXY="*.us, .org"`.
- `NO_PROXY="*"` disables all proxies.
- `NO_PROXY` with an empty list disables no proxies.

Cloud `NO_PROXY` Usage

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

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

Where Proxies Apply

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

Destinations

- S3 Compatible Stores
- AWS Kinesis Streams
- AWS CloudWatch Logs
- AWS SQS

- Azure Blob Storage
- Azure Monitor Logs
- Elasticsearch
- Honeycomb
- Splunk HEC

Sources

- AWS Kinesis Streams
- AWS SQS
- AWS S3

Collectors

- S3 Collector

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.

;

11.7.5. SSO/Okta Configuration

If you are a Cribl Stream admin and want to offer single sign-on (SSO) to your Cribl Stream users, you first choose OpenID Connect as the authentication type, then choose an SSO provider for OpenID Connect. Once configuration is complete (several steps later), the Cribl Stream login page will send users to the SSO provider login page.

The provider can be Okta or Google, among others. This page describes how to configure SSO with Okta as the provider. **SSO with Okta is supported only in Cribl Stream (LogStream) versions 3.0.0 and newer.**

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:

- A Cribl Stream Role can map to more than one Okta group.
- An Okta group can map to more than one Cribl Stream Role.

  The example mappings below illustrate these principles. Clearly, the groups in Mapping **b** and **c** each map to multiple Roles. And 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`

- 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.

Integrate Okta with Cribl Stream

1. Log in to your Okta tenant admin console.

2. In the left menu, 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 integration name** field, enter `Cribl Stream .`

   - (Optional:) In the **Logo** field, upload the Cribl logo. You can use a logo from the Cribl Media Kit.

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.

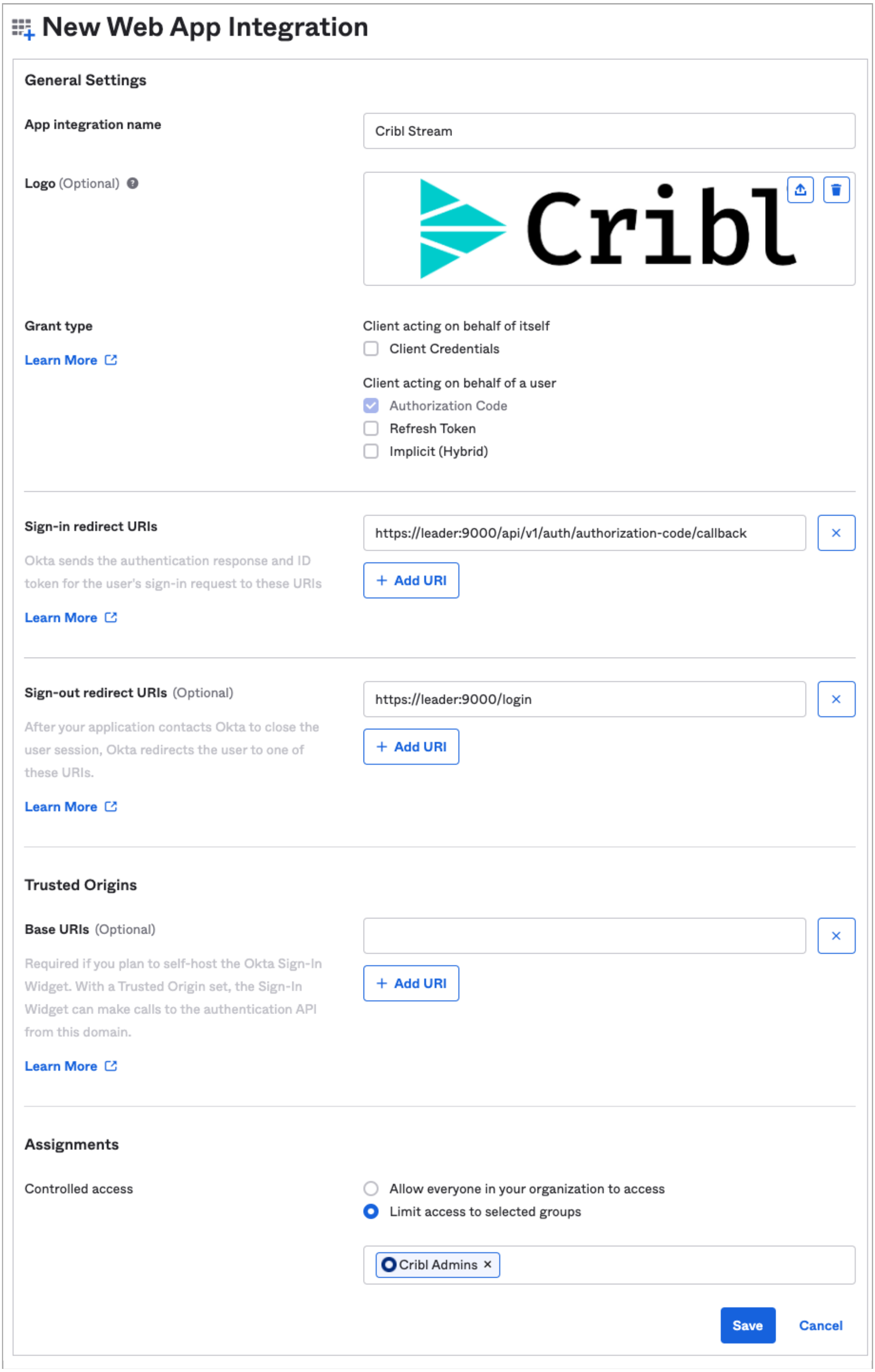
6. (Optional) In the **Sign-out redirect URIs** field, append `/login` to the pre-filled path.

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

☑ Authorization Code

□ Refresh Token

□ Implicit (Hybrid)

Sign-in redirect URIs

Okta sends the authentication response and ID token for the user's sign-in request to these URIs

Learn More

https://leader:9000/api/v1/auth/authorization-code/callback

+ Add URI

Sign-out redirect URIs (Optional)

After your application contacts Okta to close the user session, Okta redirects the user to one of these URIs.

Learn More

https://leader:9000/login

+ Add URI

Trusted Origins

Base URIs (Optional)

Required if you plan to self-host the Okta Sign-In Widget. With a Trusted Origin set, the Sign-In Widget can make calls to the authentication API from this domain.

Learn More

+ Add URI

Assignments

Controlled access

○ Allow everyone in your organization to access

● Limit access to selected groups

Cribl Admins ×

Save Cancel

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.

Configure Cribl Stream

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:

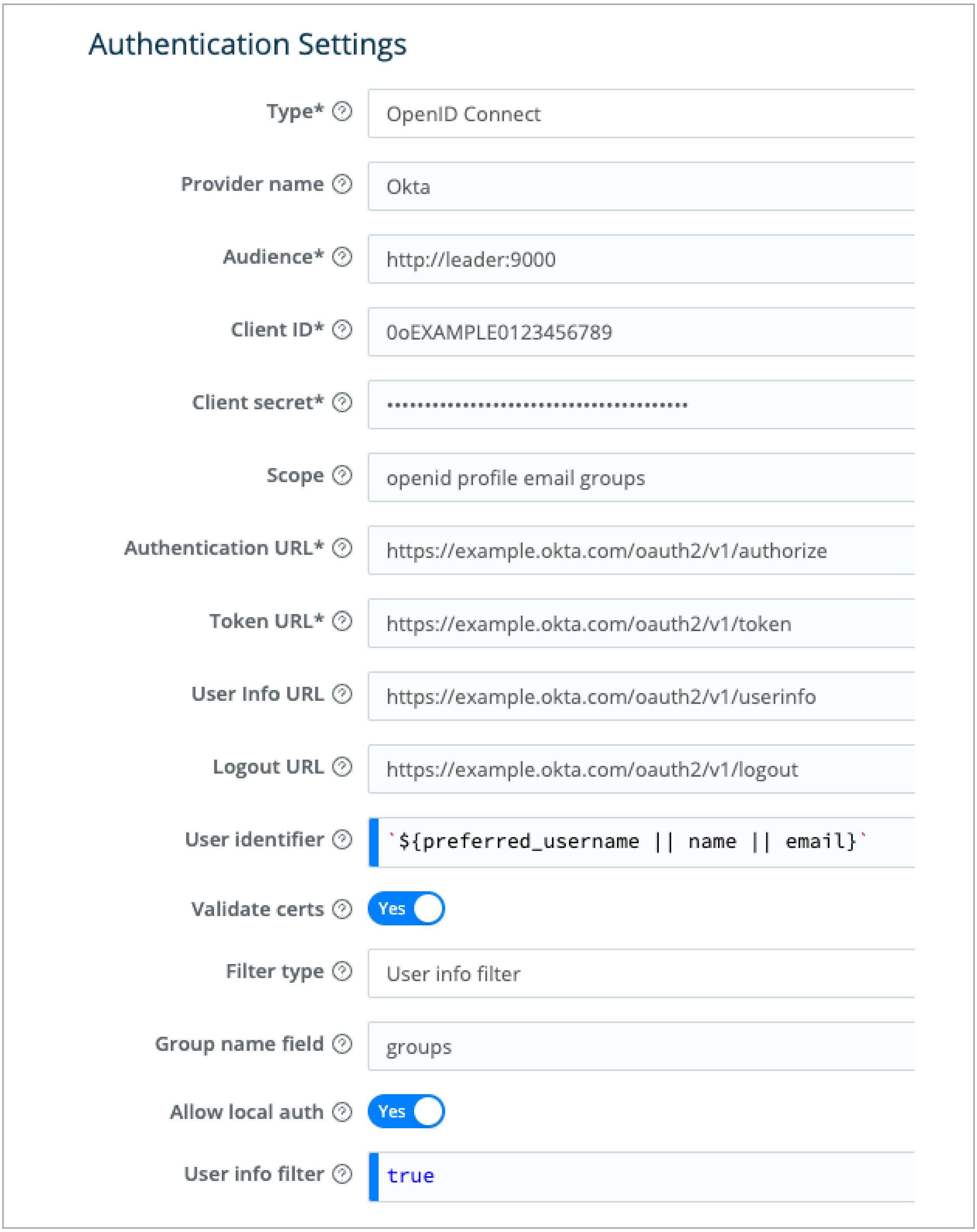
     ```
     {
       "auth": "https://dev-416897.oktapreview.com/oauth2/v1/authorize",
       "token": "https://dev-416897.oktapreview.com/oauth2/v1/token",
       "userinfo": "https://dev-416897.oktapreview.com/oauth2/v1/userinfo",
       "logout": "https://dev-416897.oktapreview.com/oauth2/v1/logout"
     }
     ```

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 (e.g., 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.

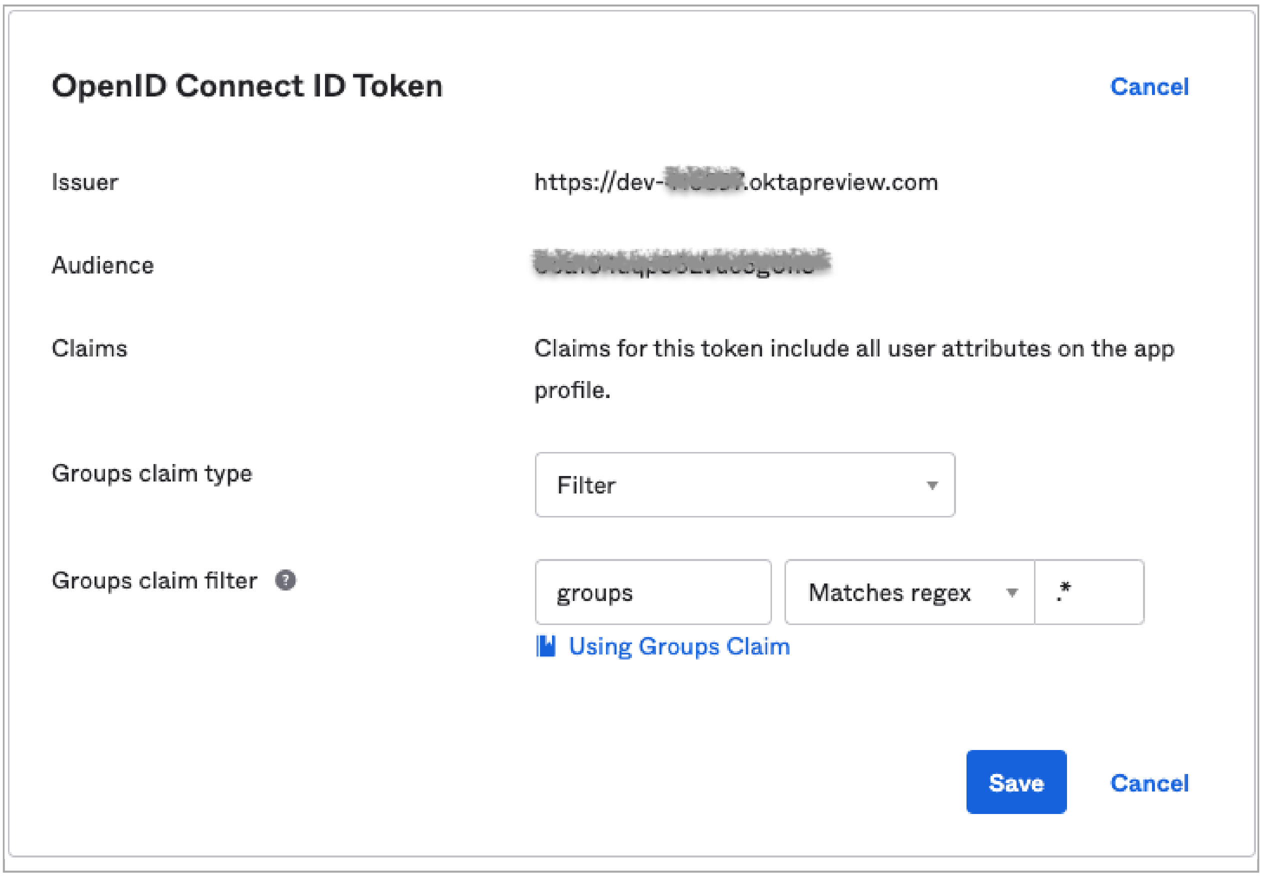
In Okta, you should still be in the panel for the app you created. If not, you can get there by opening **Applications** > **Applications** and selecting the app.

- 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.

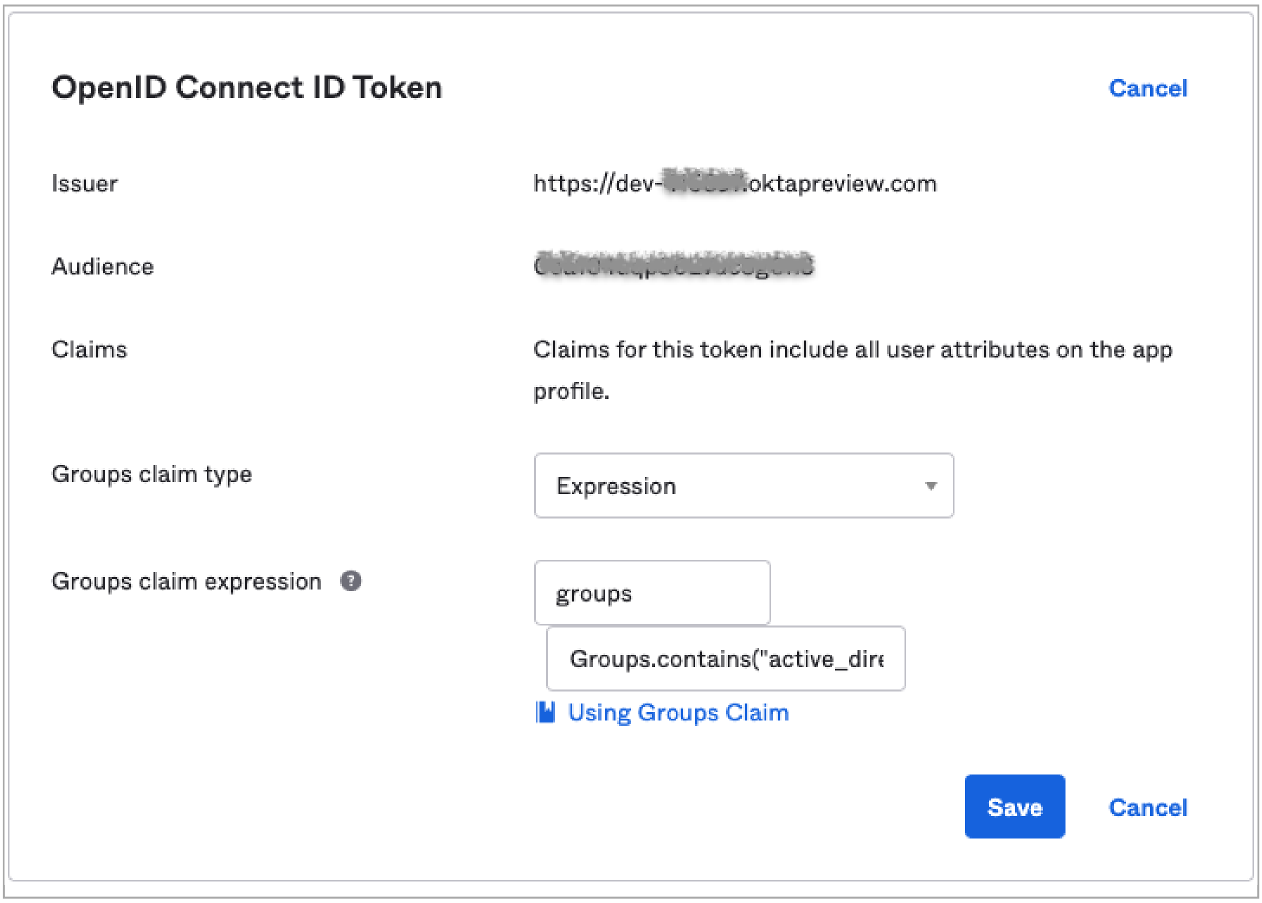
2. Set **Groups claim type** set to `Expression.`

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

Cancel

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.

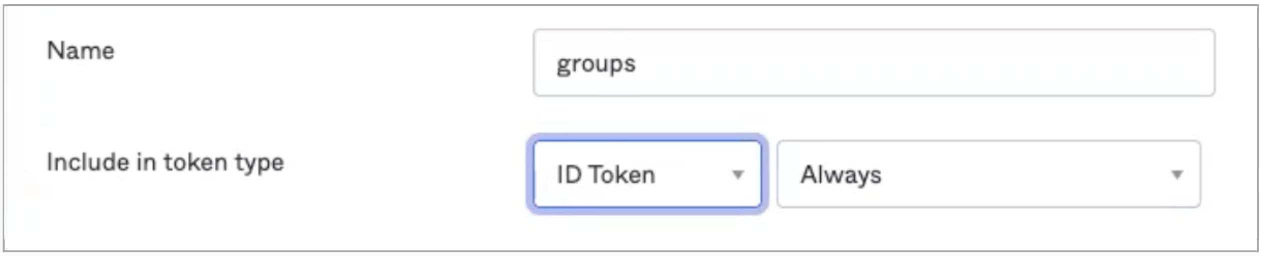
1. In Okta, open the **Security** > **API** page.

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).

Map Okta Group Names to Cribl Stream Roles

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.

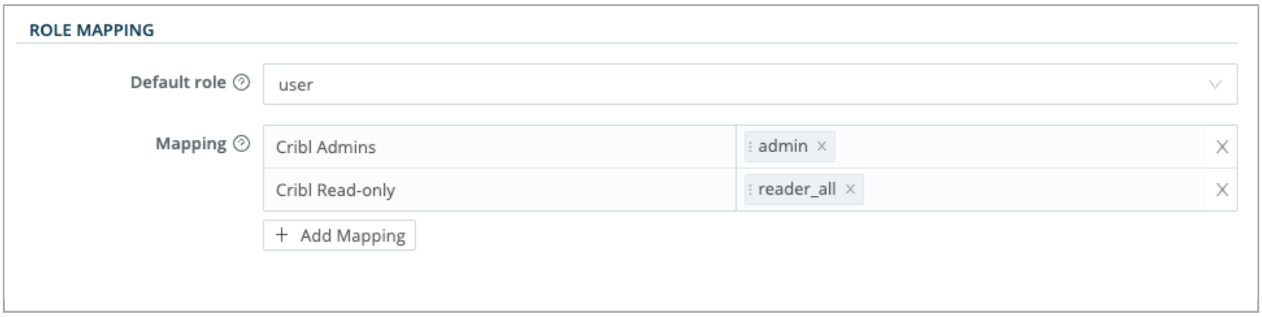
1. In Cribl Stream, select **Settings** > **Access Management** > **Authentication**.

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).



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.

;

12. Setup Guides

The following topics provide expanded configuration options for selected Cribl Stream integrations (Collectors, Sources, Destinations, and Notifications):

- Cribl Edge to Cribl Stream
- Configuring Upstream Logging Agents
- Azure AD + OpenID Configuration
- Azure Event Hubs Integrations
- Splunk Cloud and BYOL Integrations
- Webhook/BigPanda Integration
- Webhook/Sumo Logic Integration
- Zscaler NSS Integration

;

12.1. Cribl Edge to Cribl Stream

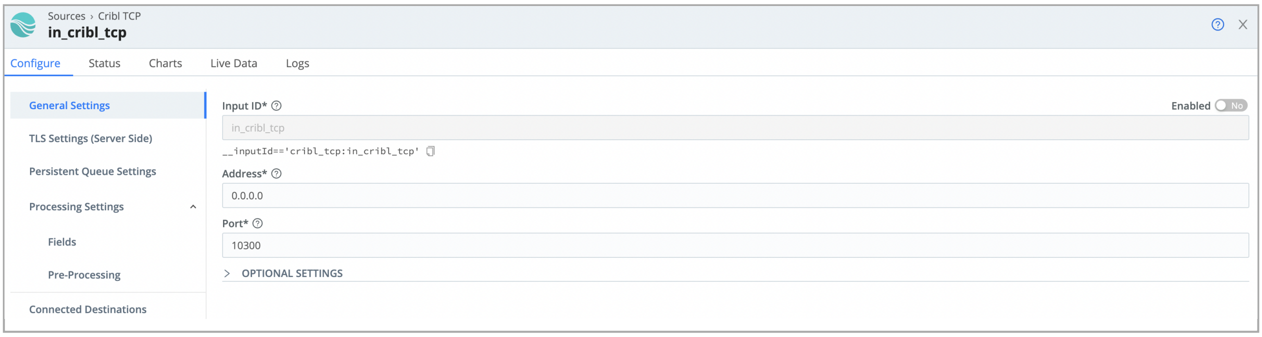
Cribl Edge automatically discovers logs, metrics, application data, etc. – in real time – from your configured endpoints, and delivers them to Cribl Stream or any supported destination. Meanwhile, Cribl Stream can help collect, reduce, enrich, transform, and route data from Cribl Edge to any destination. And using a Cribl TCP Source, you can collect and route data from Edge Nodes to Stream Worker Nodes connected to the same Leader, without incurring additional cost.

This guide outlines how to route data from an Edge Node (or an entire Fleet) to an existing Stream Worker Group for additional processing, using the Cribl TCP Source and Destination.

> 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

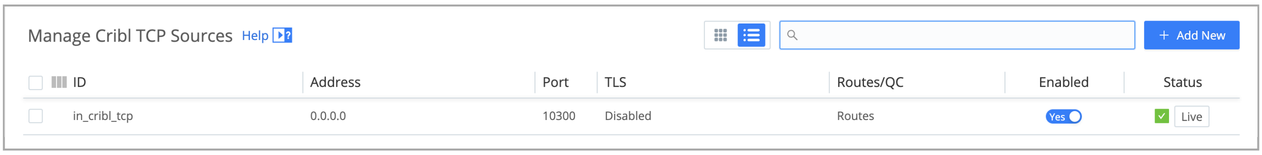
`__inputId=='cribl_tcp:in_cribl_tcp'`

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.


Manage Cribl TCP Sources Help

ID	Address	Port	TLS	Routes/QC	Enabled	Status
in_cribl_tcp	0.0.0.0	10300	Disabled	Routes	Yes	✓ Live

Status of the Cribl TCP Source

Configure the Exec Source on Cribl Edge

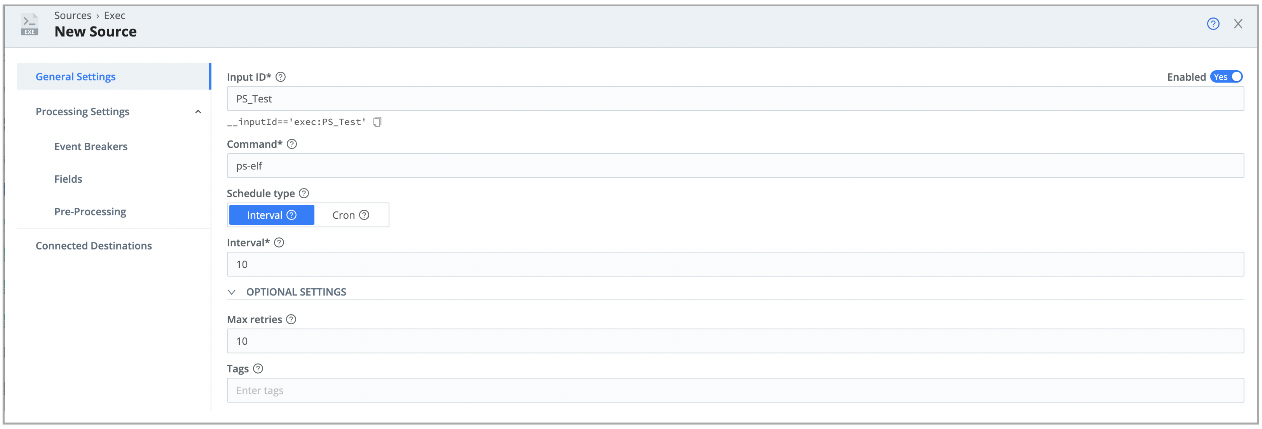
Next, we'll configure the Exec Source on your Edge Node. This Source will break the incoming streams of data into discrete events, and send them to Cribl Stream.

> 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.

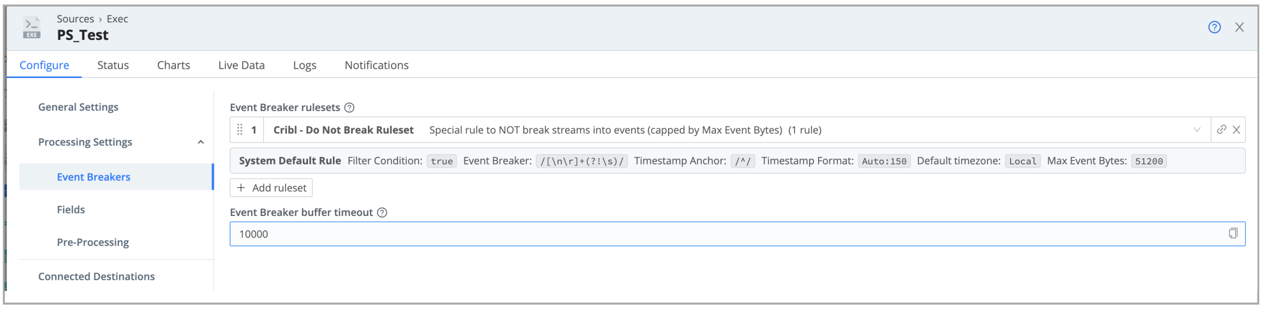

Sources › Exec
New Source

Input ID*: PS_Test
`__inputId=='exec:PS_Test'`
Enabled: Yes
Command*: ps-elf
Schedule type: Interval
Interval*: 10
OPTIONAL SETTINGS
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`.



Event Breaker rulesets

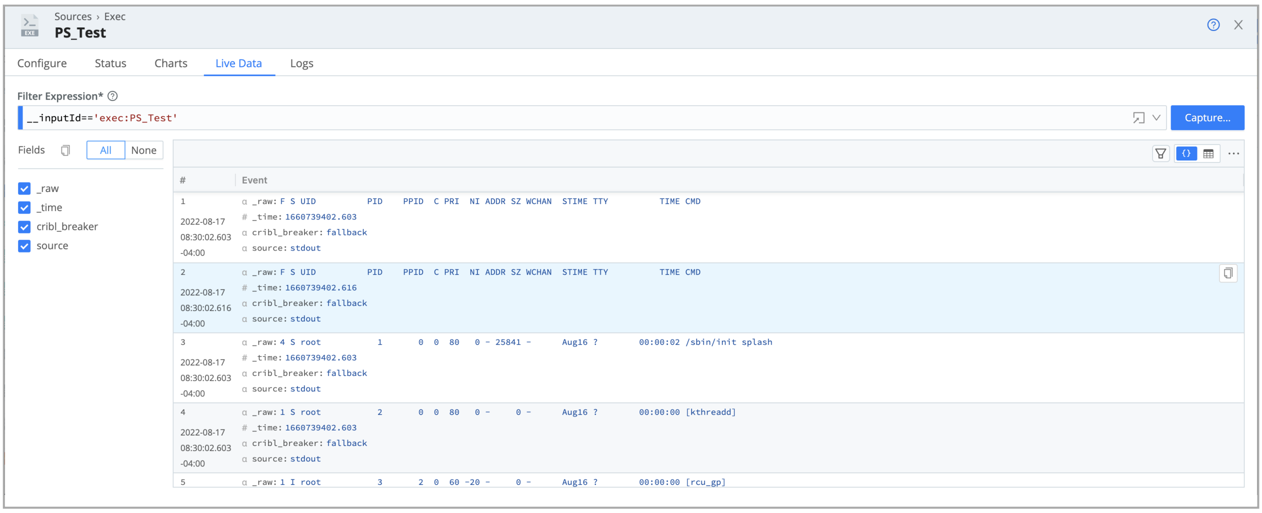
1 **Cribl - Do Not Break Ruleset** Special rule to NOT break streams into events (capped by Max Event Bytes) (1 rule)

System Default Rule Filter Condition: `true` Event Breaker: `/[\n\r]+(?!\s)/` Timestamp Anchor: `/^/` Timestamp Format: `Auto:150` Default timezone: `Local` Max Event Bytes: `51200`

Event Breaker buffer timeout: 10000

Apply an Event Breaker

Next, preview your data on the modal's **Live Data** tab.



Filter Expression*: `__inputId=='exec:PS_Test'`

```
#  Event
1  _raw: F S UID         PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
   2022-08-17 08:30:02.603 -04:00
2  _raw: F S UID         PID    PPID  C PRI  NI ADDR SZ WCHAN  STIME TTY          TIME CMD
   _time: 1660739402.616
   cribl_breaker: fallback
   source: stdout
   2022-08-17 08:30:02.616 -04:00
3  _raw: 4 S root          1       0  0  80   0 - 25841 -      Aug16 ?        00:00:02 /sbin/init splash
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
   2022-08-17 08:30:02.603 -04:00
4  _raw: 1 S root          2       0  0  80   0 -     0 -      Aug16 ?        00:00:00 [kthreadd]
   _time: 1660739402.603
   cribl_breaker: fallback
   source: stdout
   2022-08-17 08:30:02.603 -04:00
5  _raw: 1 I root          3       2  0  60 -20 -     0 -      Aug16 ?        00:00:00 [rcu_gp]
```

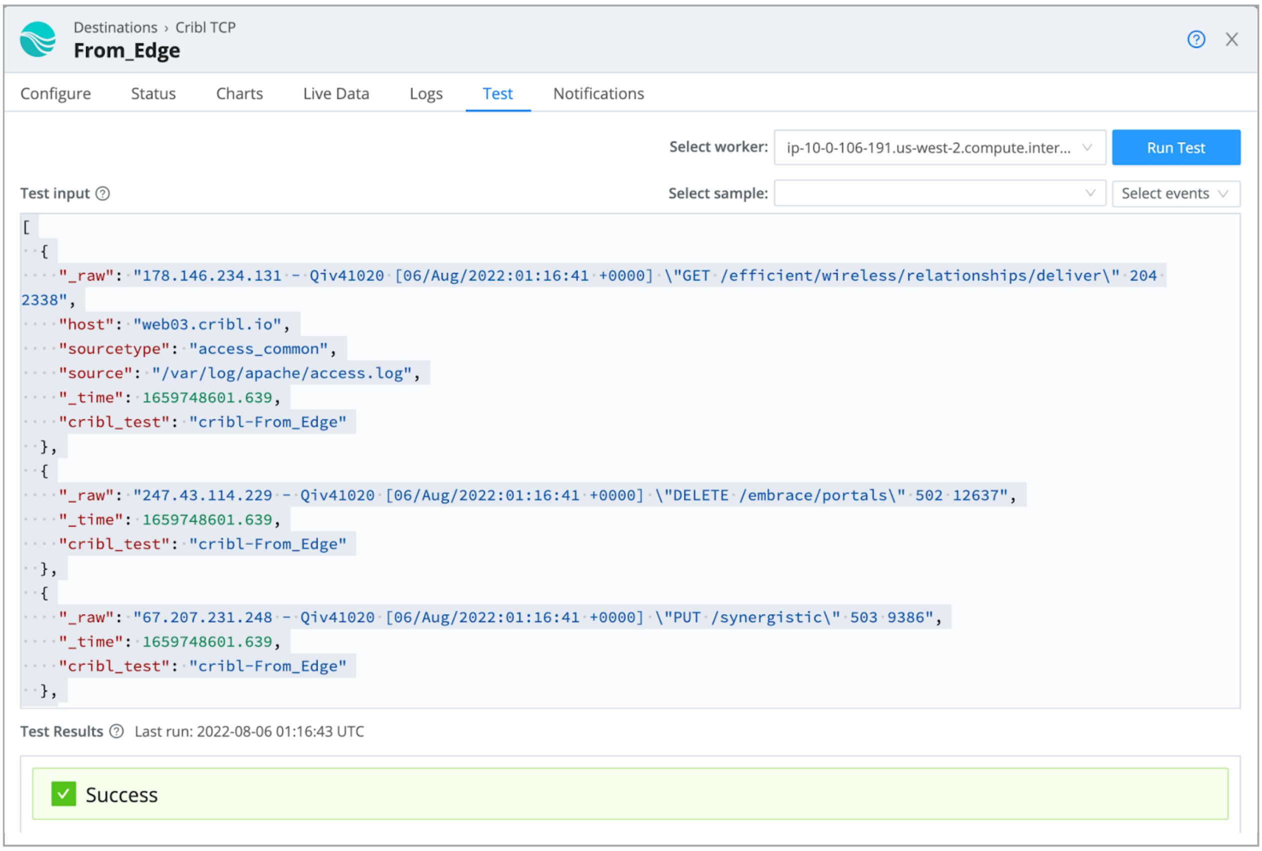
Preview Live Data

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.

Once you've configured your Destination, test it to verify that your Edge Node can communicate with the Stream Worker Group.



```
[
  {
    "_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"
  },
```

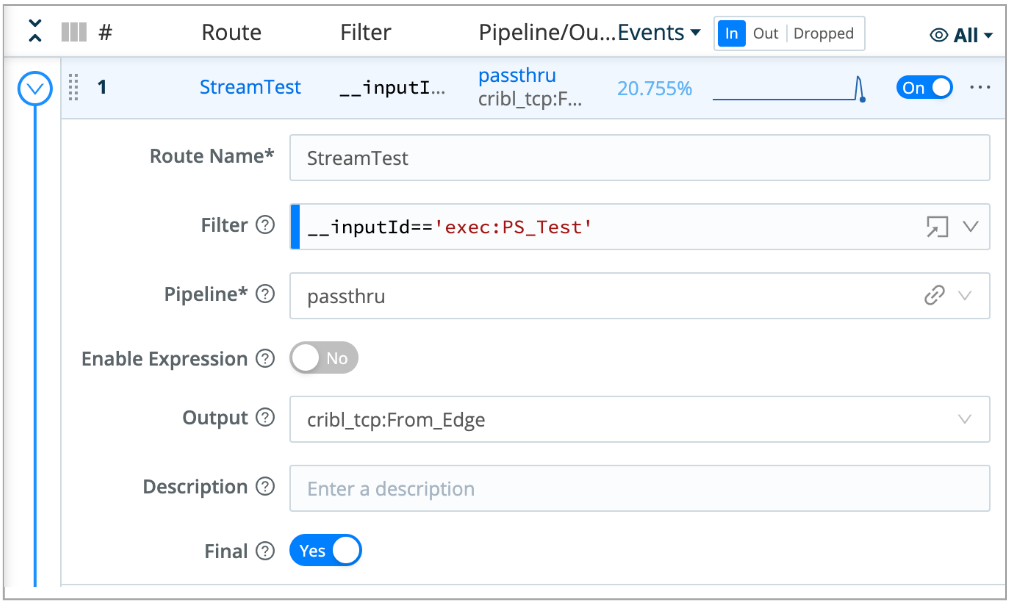
Test Results Last run: 2022-08-06 01:16:43 UTC

☑ 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 Name*: StreamTest

Filter: `__inputId=='exec:PS_Test'`

Pipeline*: passthru

Enable Expression: No

Output: cribl_tcp:From_Edge

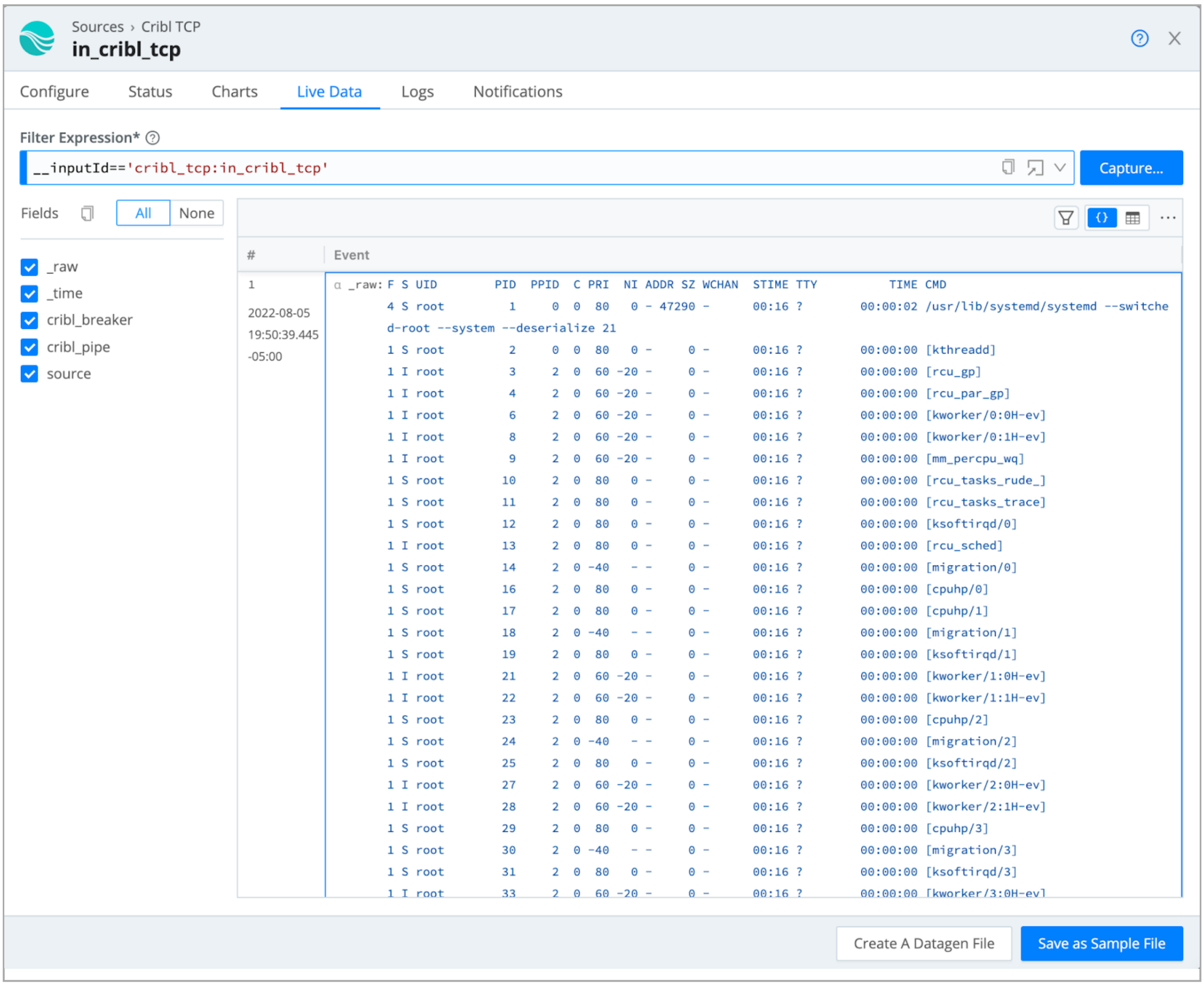
Description: Enter a description

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'
```

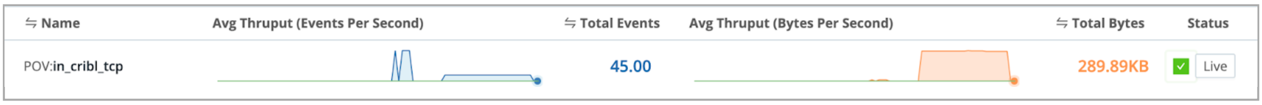
1

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

;

12.2. 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** 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.logstream.<cloud_instance>.cribl.cloud
    port         8088
    splunk_token <HEC_token>
    tls          on
    # optional
    event_source logs_from_fluentd
```

Save the changes and restart the `td-agent-bit` service.

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** 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.logstream.<cloud_instance>.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. This setting is in Cribl Stream's **Sources** > **Splunk TCP** > **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.logstream.<cloud_instance>.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

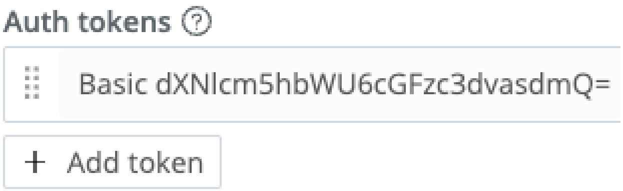
Cribl Stream can ingest the native Elasticsearch streaming protocol directly.

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:



Auth tokens

Basic dXNlcm5hbWU6cGFzc3dvasdmQ=

Basic auth in the Cribl Stream Elasticsearch API Source

```
output.elasticsearch:
  hosts: ["in.logstream.<cloud_instance>.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.logstream.<cloud_instance>.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.logstream.<cloud_instance>.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-
edition/3.33/administration-guide/34
    elasticsearch-http(url("https://worker1:9200/_bulk" "https://worker2:9200/_bulk"
"https://worker3:9200/_bulk")
        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.logstream.<cloud_instance>.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.logstream.<cloud_instance>.cribl.cloud"
          port="9514"
          protocol="tcp"
          action.resumeRetryCount="100"
          queue.type="linkedList"
          queue.size="10000"
)
```

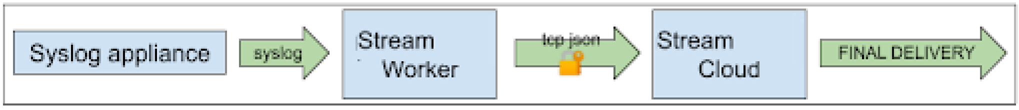
Save the changes and restart the rsyslog service.

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.logstream.<cloud_instance>.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.logstream.<cloud_instance>.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.logstream.<cloud_instance>.cribl.cloud:10091 scope nginx
```

;

12.3. Azure AD + OpenID Configuration

This page outlines how to integrate Azure Active Directory with Cribl Stream's SSO/OpenID Connect authentication.

Configure Azure AD App

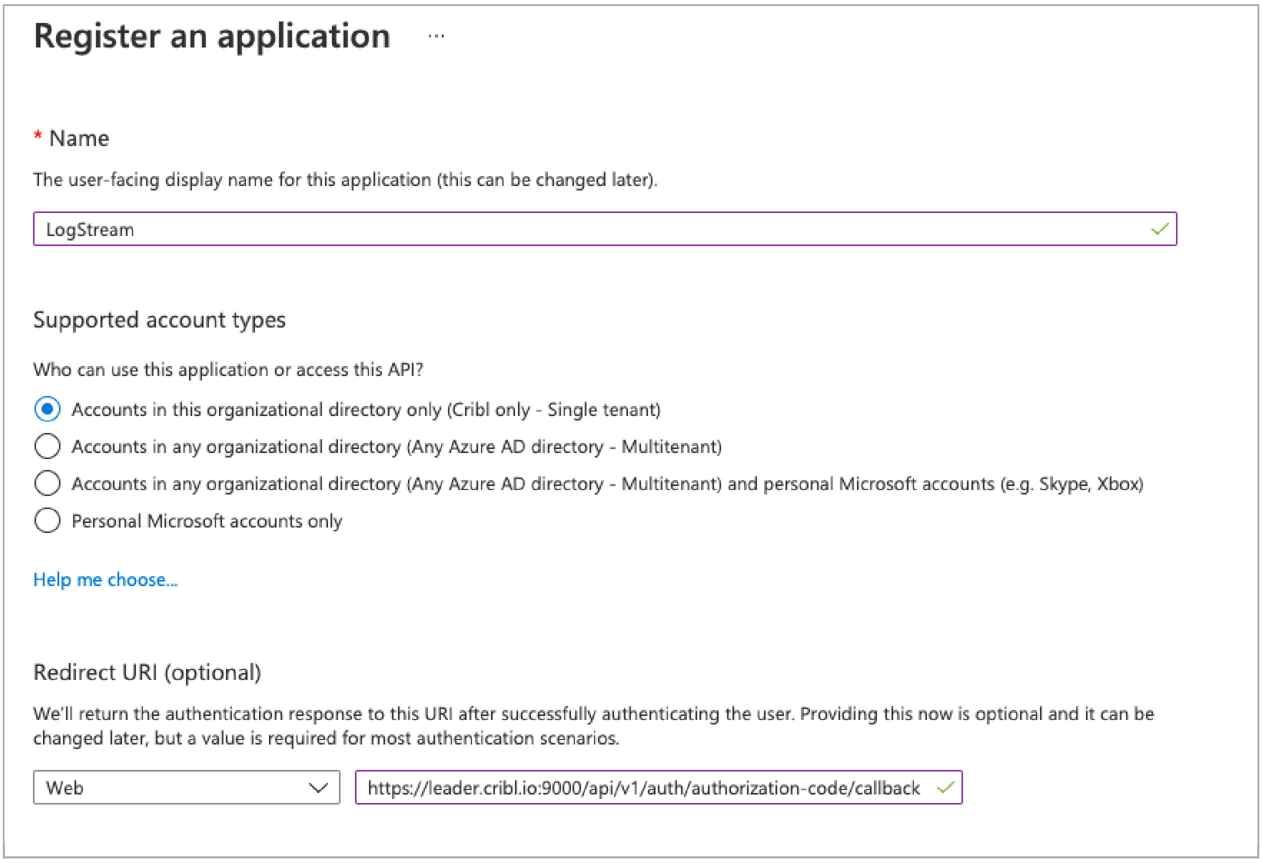
Start at the Azure portal to configure an OpenID Connect provider: https://portal.azure.com/.

Register Your Azure AD App

1. Open the **Azure Active Directory** 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).

LogStream ✓

Supported account types

Who can use this application or access this API?

● Accounts in this organizational directory only (Cribl 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.

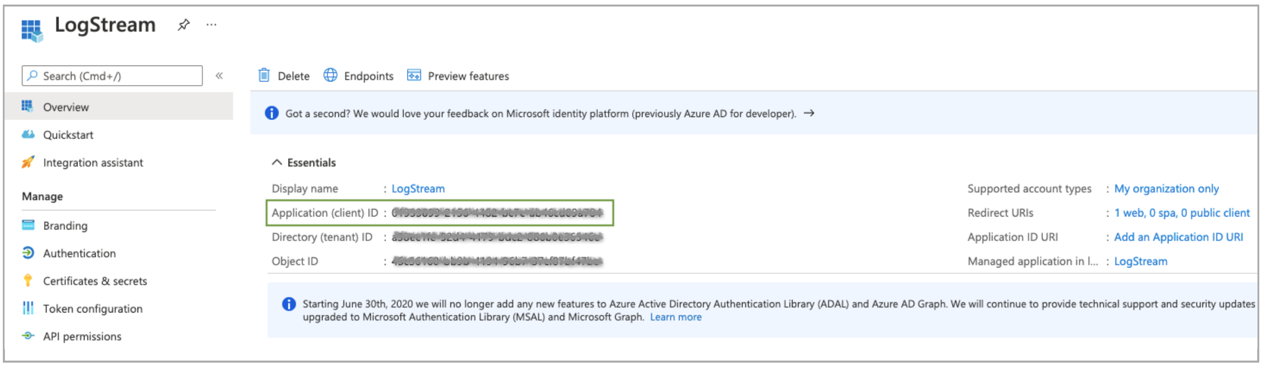
Web | https://leader.cribl.io:9000/api/v1/auth/authorization-code/callback ✓

Registering an Azure AD app

Get the Azure AD 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**.



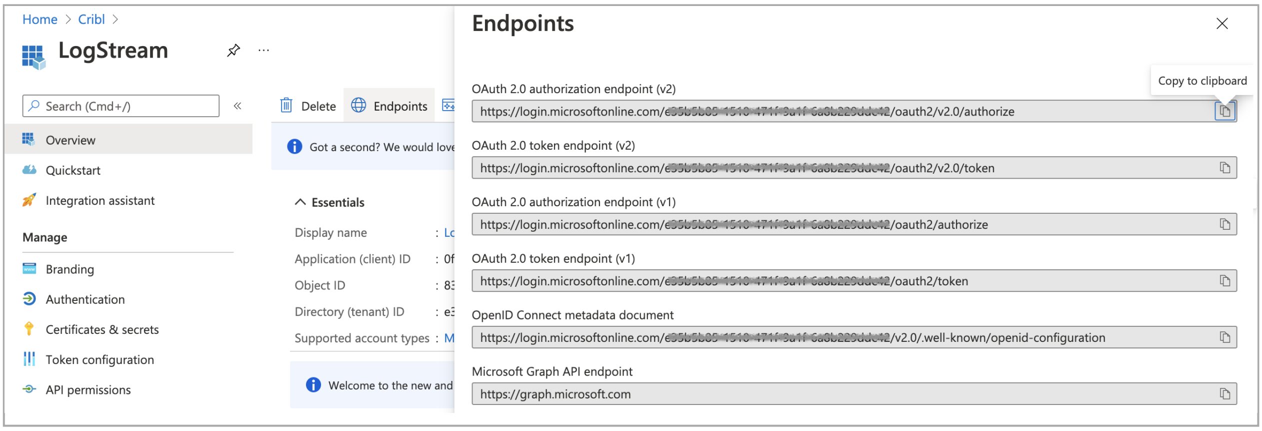
Got a second? We would love your feedback on Microsoft identity platform (previously Azure AD for developer). →

Display name : LogStream
Application (client) ID : [redacted]
Directory (tenant) ID : [redacted]
Object ID : [redacted]
Supported account types : My organization only
Redirect URIs : 1 web, 0 spa, 0 public client
Application ID URI : Add an Application ID URI
Managed application in l... : LogStream

Starting June 30th, 2020 we will no longer add any new features to Azure Active Directory Authentication Library (ADAL) and Azure AD Graph. We will continue to provide technical support and security updates
upgraded to Microsoft Authentication Library (MSAL) and Microsoft Graph. Learn more

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.



Got a second? We would love

Display name : Lo
Application (client) ID : 0f
Object ID : 83
Directory (tenant) ID : e3
Supported account types : M

Welcome to the new and

Endpoints

OAuth 2.0 authorization endpoint (v2)
https://login.microsoftonline.com/[redacted]/oauth2/v2.0/authorize

OAuth 2.0 token endpoint (v2)
https://login.microsoftonline.com/[redacted]/oauth2/v2.0/token

OAuth 2.0 authorization endpoint (v1)
https://login.microsoftonline.com/[redacted]/oauth2/authorize

OAuth 2.0 token endpoint (v1)
https://login.microsoftonline.com/[redacted]/oauth2/token

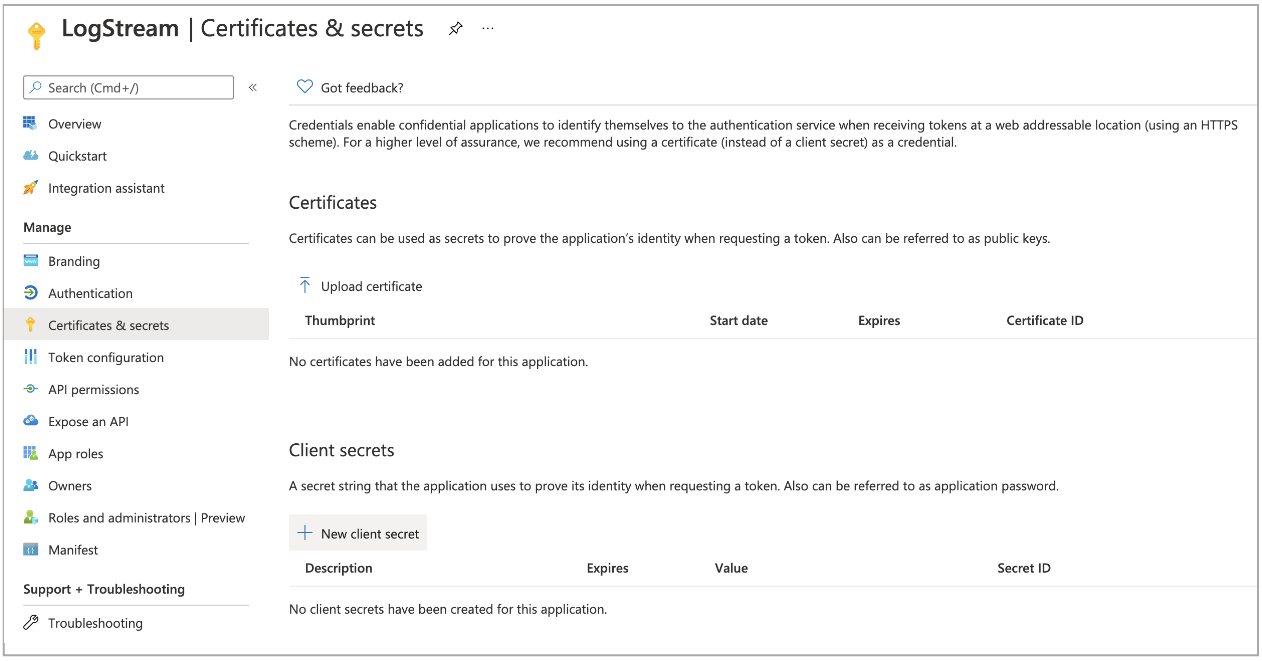
OpenID Connect metadata document
https://login.microsoftonline.com/[redacted]/v2.0/.well-known/openid-configuration

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.

Certificates

Certificates can be used as secrets to prove the application's identity when requesting a token. Also can be referred to as public keys.

Thumbprint	Start date	Expires	Certificate ID

No certificates have been added for this application.

Client secrets

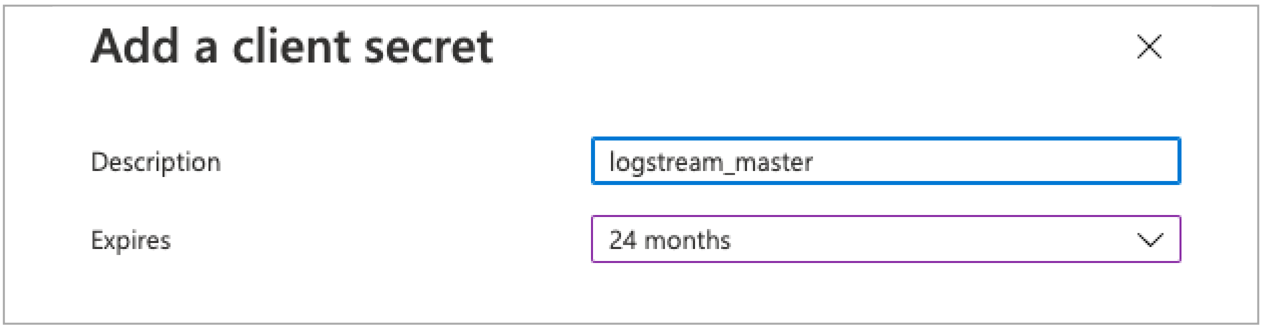
A secret string that the application uses to prove its identity when requesting a token. Also can be referred to as application password.

Description	Expires	Value	Secret ID

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

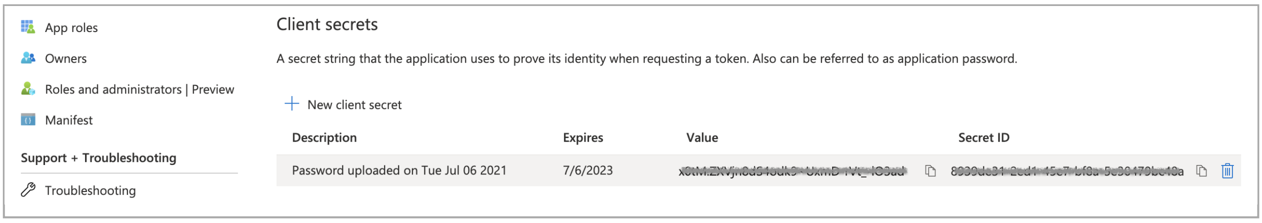
Description: logstream_master

Expires: 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.



Client secrets

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
Password uploaded on Tue Jul 06 2021	7/6/2023	[redacted]	[redacted]

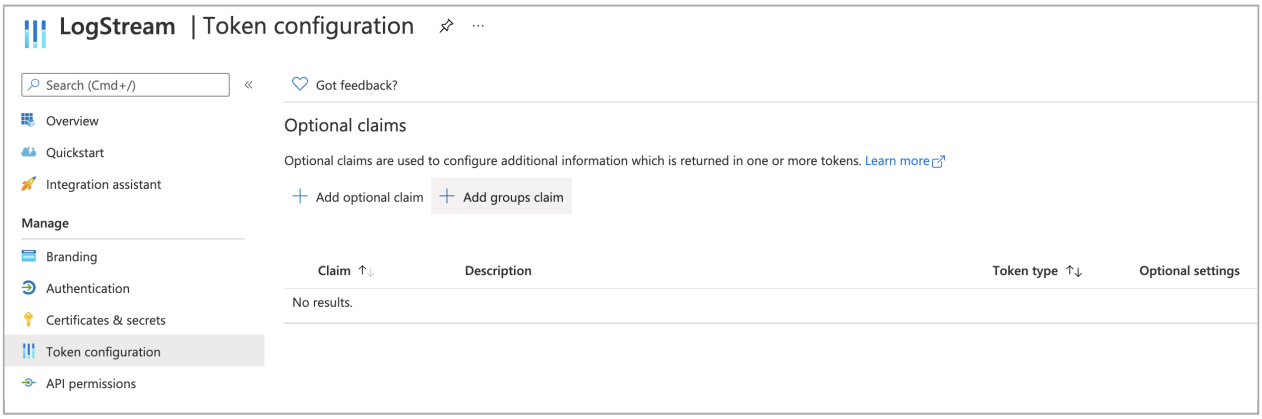
Copy that secret!

> 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**.

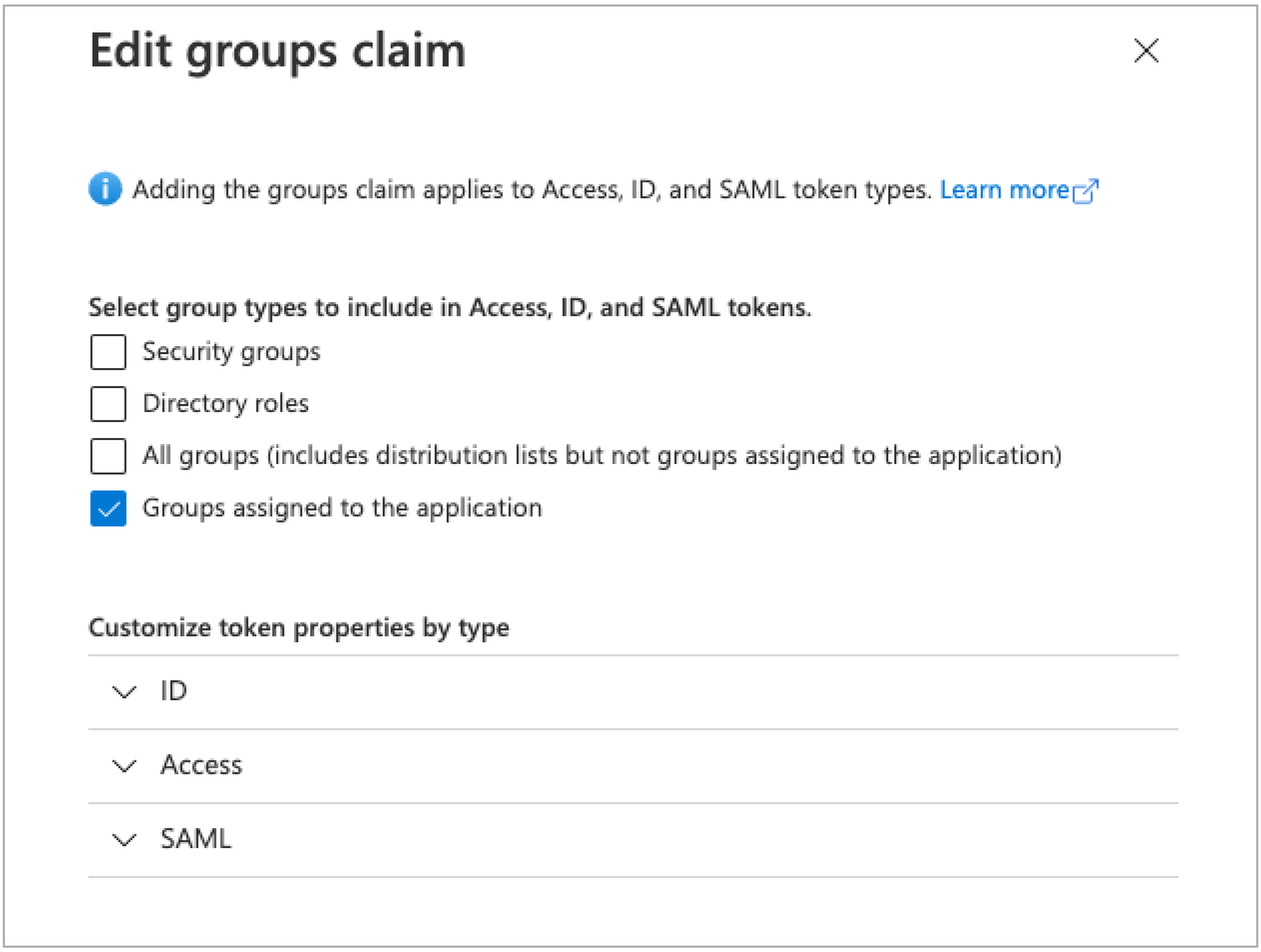


Optional claims are used to configure additional information which is returned in one or more tokens. Learn more

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.

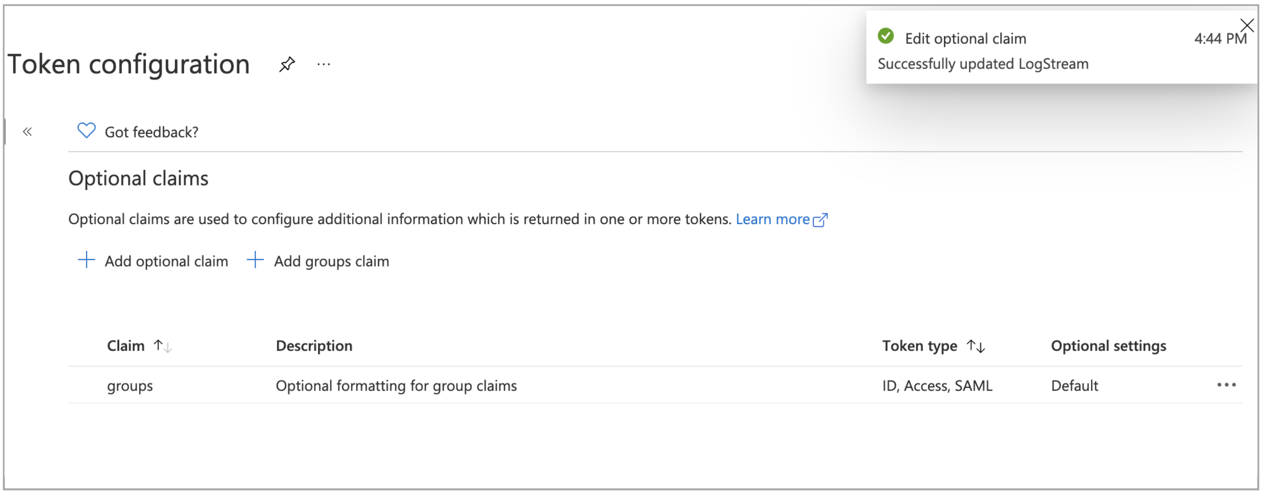
☐ Security groups
☐ Directory roles
☐ All groups (includes distribution lists but not groups assigned to the application)
☑ Groups assigned to the application

Customize token properties by type

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.

3. Your token is now configured, and you're all done on the Azure side.



Edit optional claim 4:44 PM
Successfully updated LogStream

Optional claims

Optional claims are used to configure additional information which is returned in one or more tokens. Learn more

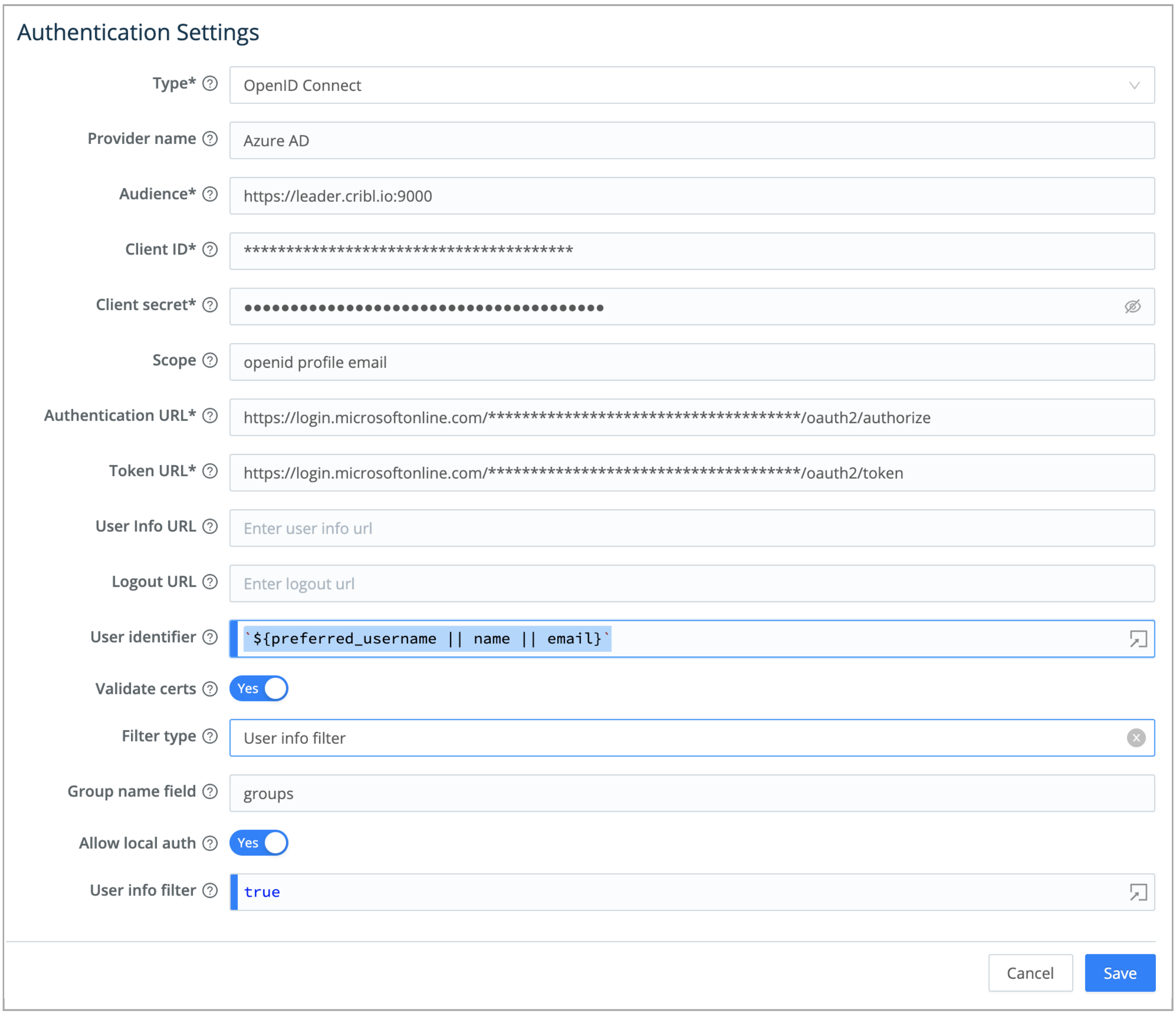
Claim	Description	Token type	Optional settings
groups	Optional formatting for group claims	ID, Access, SAML	Default

Azure AD token configuration complete

Configure Cribl Stream Authentication

Switch to Cribl Stream, and navigate to its global ⚙ **Settings** (lower left) > **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`
- **Client ID**: Enter you r Azure AD **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

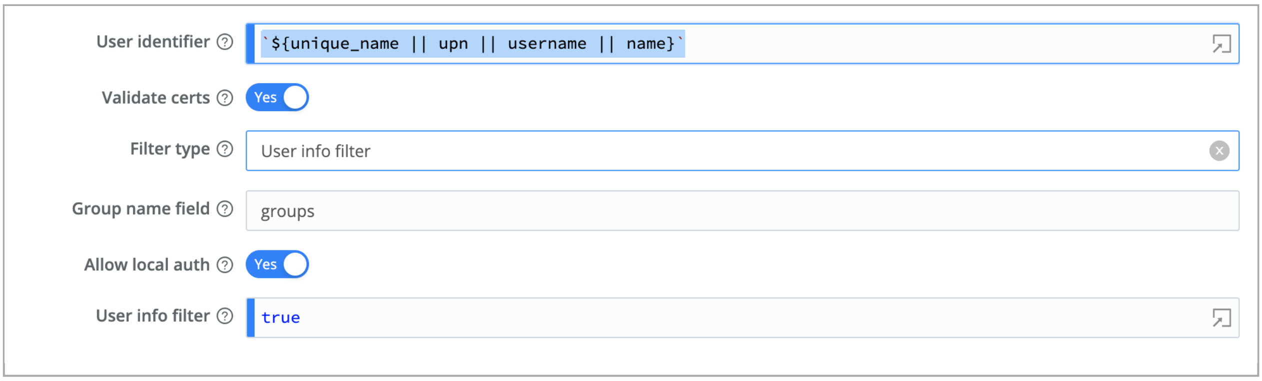
Filter type User info filter

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

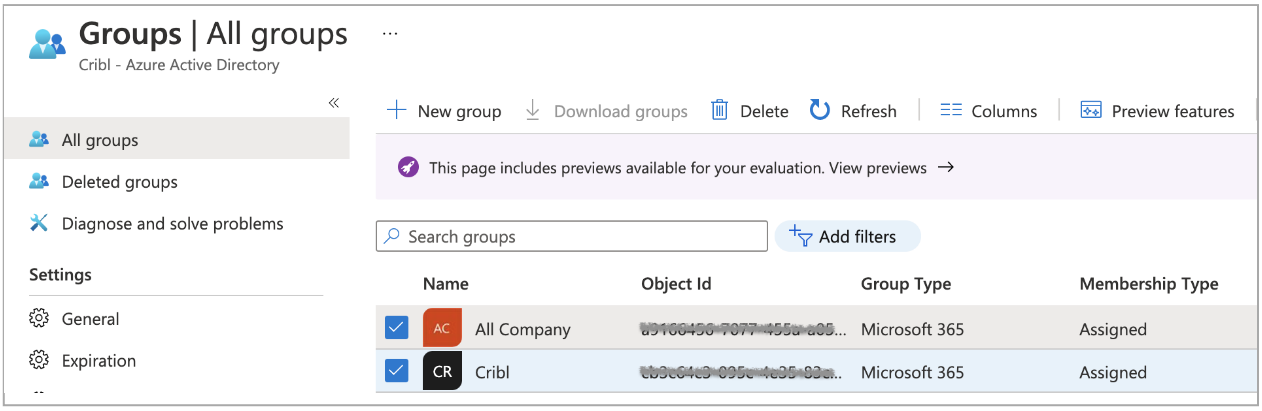
User info filter `true`

Sample User identifier entry for v1 endpoints

Map Azure AD Groups to Cribl Stream Roles

Next, map your Azure AD groups to Cribl Stream Roles. The group names might appear as GUIDs. You can translate these on the Azure AD **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.



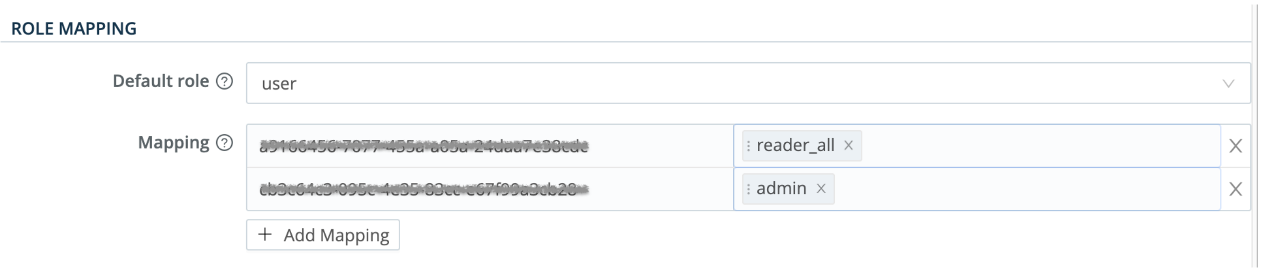
Groups | All groups

Cribl - Azure Active Directory

This page includes previews available for your evaluation. View previews →

	Name	Object Id	Group Type	Membership Type
☑ AC	All Company	[redacted]	Microsoft 365	Assigned
☑ CR	Cribl	[redacted]	Microsoft 365	Assigned

Azure AD groups...



ROLE MAPPING

Default role: user

Mapping	
[redacted]	reader_all
[redacted]	admin

...mapped to Cribl Stream Roles

;

12.4. Azure Event Hubs Integrations

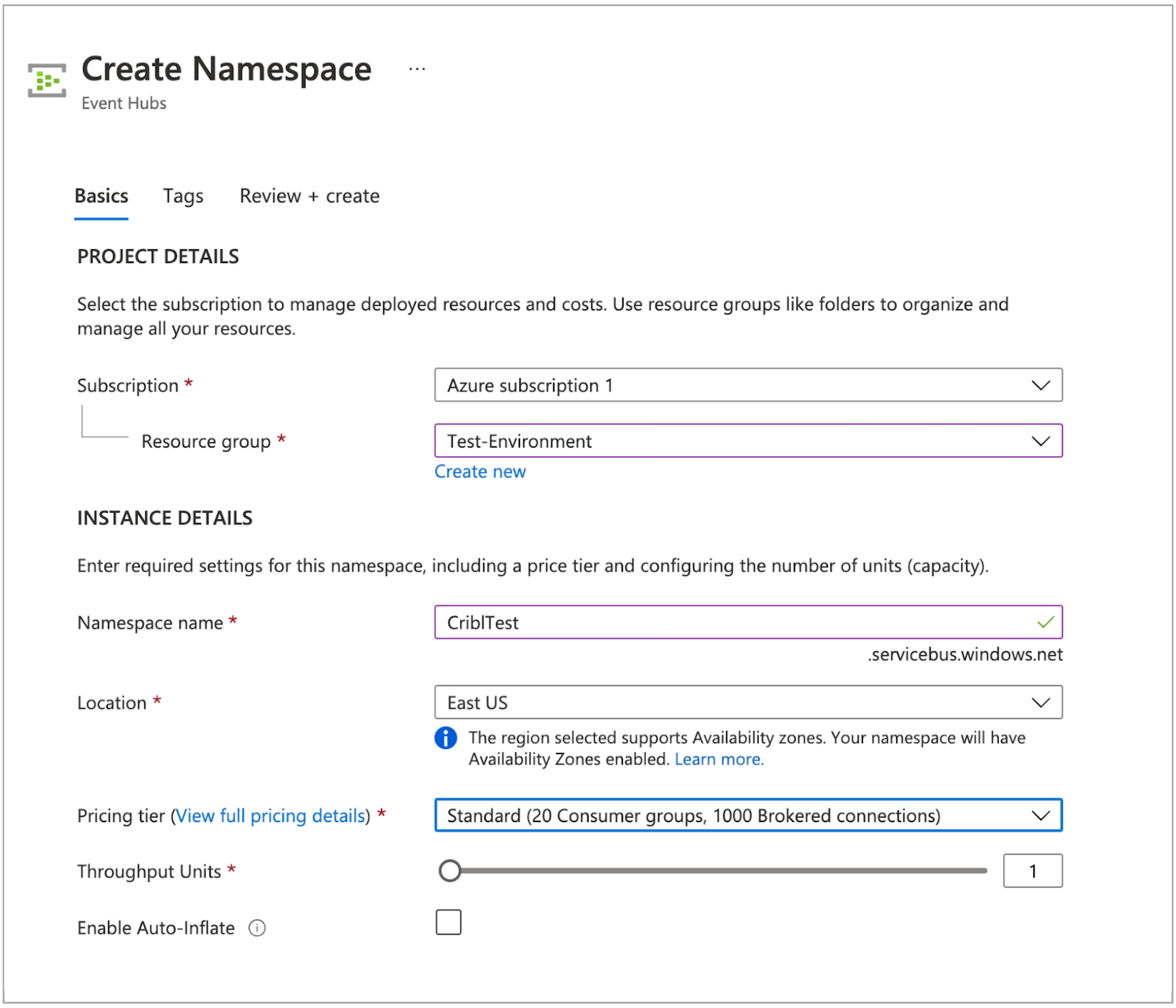
You can create an Azure Event Hub which sends data to a Cribl Stream Azure Event Hubs Source.

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.

For purposes of this tutorial, we assume that in the **Create Namespace** page, you will use `CriblTest` as your **Namespace name**.



Create Namespace ···
Event Hubs

Basics | Tags | Review + create

PROJECT DETAILS

Select the subscription to manage deployed resources and costs. Use resource groups like folders to organize and manage all your resources.

Subscription * : Azure subscription 1
Resource group * : Test-Environment
Create new

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
The region selected supports Availability zones. Your namespace will have Availability Zones enabled. Learn more.

Pricing tier (View full pricing details) * : Standard (20 Consumer groups, 1000 Brokered connections)

Throughput Units * : 1

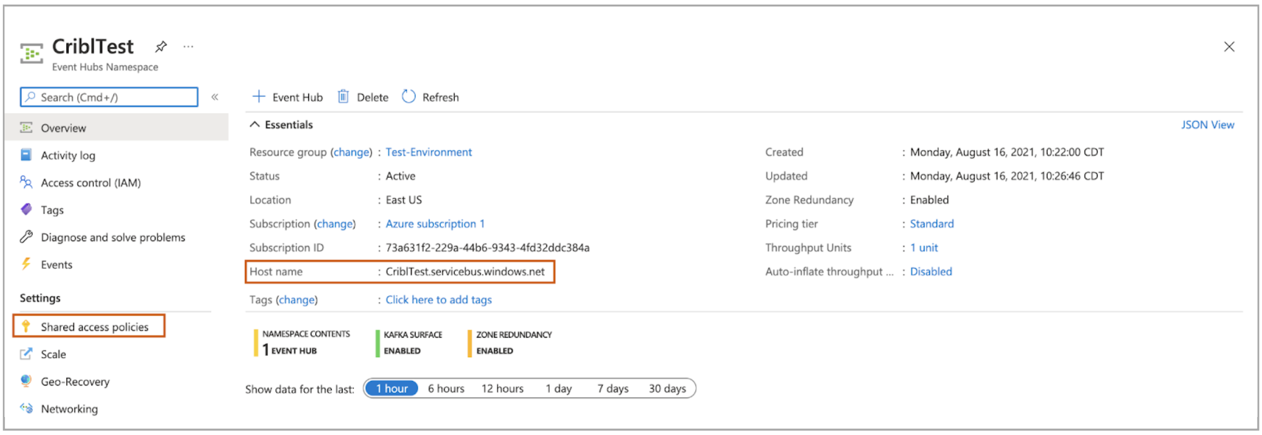
Enable Auto-Inflate ⓘ : □

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. Write down the **Host Name**.



Resource group (change) : Test-Environment
Status : Active
Location : East US
Subscription (change) : Azure subscription 1
Subscription ID : 73a631f2-229a-44b6-9343-4fd32ddc384a
Host name : CriblTest.servicebus.windows.net
Tags (change) : Click here to add tags

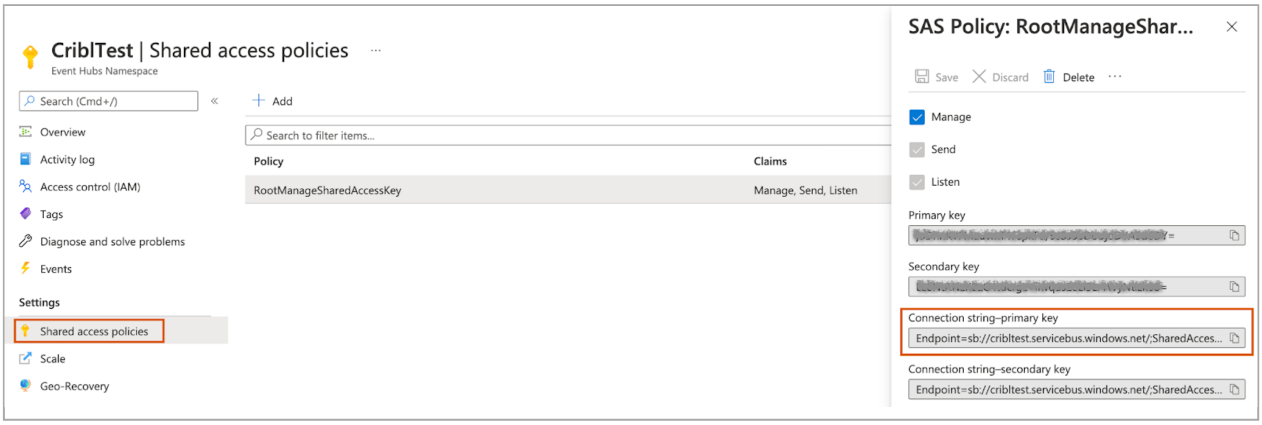
Created : Monday, August 16, 2021, 10:22:00 CDT
Updated : Monday, August 16, 2021, 10:26:46 CDT
Zone Redundancy : Enabled
Pricing tier : Standard
Throughput Units : 1 unit
Auto-inflate throughput ... : Disabled

NAMESPACE CONTENTS 1 EVENT HUB
KAFKA SURFACE ENABLED
ZONE REDUNDANCY ENABLED

Show data for the last: 1 hour 6 hours 12 hours 1 day 7 days 30 days

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/;SharedAcces...

Connection string–secondary key
Endpoint=sb://cribltest.servicebus.windows.net/;SharedAcces...

Viewing Shared access policies

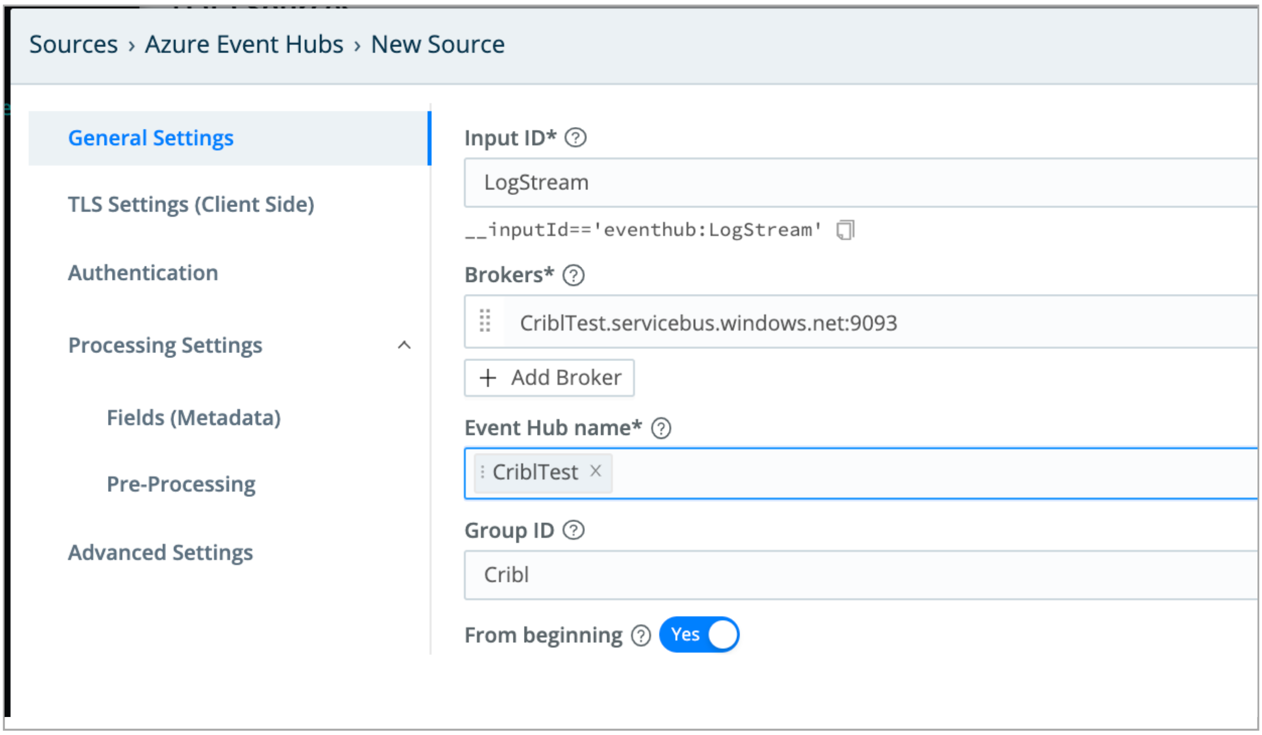
4. Copy and securely store the **Connection String-primary key**.

Configure the Azure Event Hubs Source in Cribl Stream

1. From the top nav of a Cribl Stream instance or Group, select **Data** > **Sources**, then select **Azure Event Hubs** from the **Manage Sources** page's tiles or left nav. Click **+ Add New** to open the **Azure Event Hubs** > **New Source** modal.

2. In the **General Settings** tab, enter the following values:

    - **InputId**: `LogStream`.
    - **Brokers**: To the Host Name you wrote down earlier, append port 9093, and enter the result. For example: `CriblTest.servicebus.windows.net:9093`.
    - **Event Hub Name**: The name of your Azure Event Hub, for example: `CriblTest`. This is equivalent to a Kafka topic.
    - **Group ID**: Keep (or change, if desired) the default value (`Cribl`).



Sources › Azure Event Hubs › New Source

Input ID*: LogStream

`__inputId=='eventhub:LogStream'`

Brokers*: CriblTest.servicebus.windows.net:9093

Event Hub name*: CriblTest

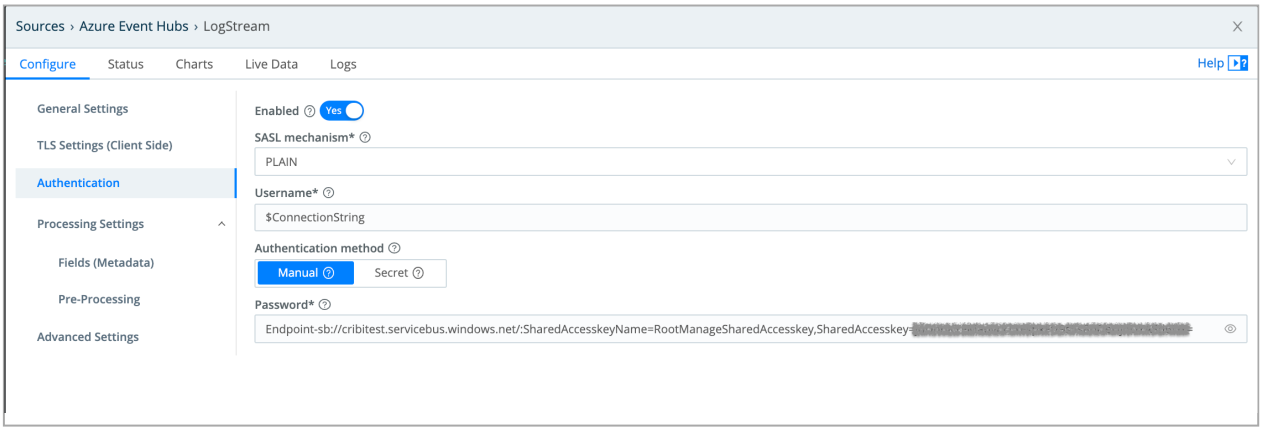
Group ID: Cribl

From beginning: Yes

The General Settings tab

6. In the **Authentication** tab, enter or select the following values:

    - **SASL mechanism**: `PLAIN` (the only supported option).
    - **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.



Enabled: Yes

SASL mechanism*: PLAIN

Username*: $ConnectionString

Authentication method: Manual | Secret

Password*: Endpoint-sb://cribitest.servicebus.windows.net/:SharedAccesskeyName=RootManageSharedAccesskey,SharedAccesskey=[redacted]

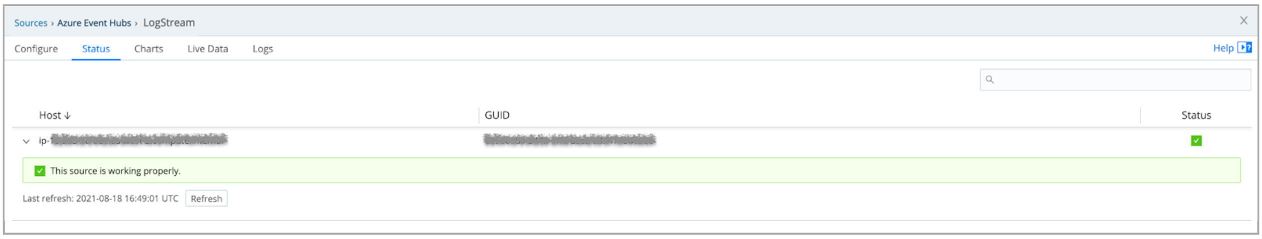
The Authentication tab

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.



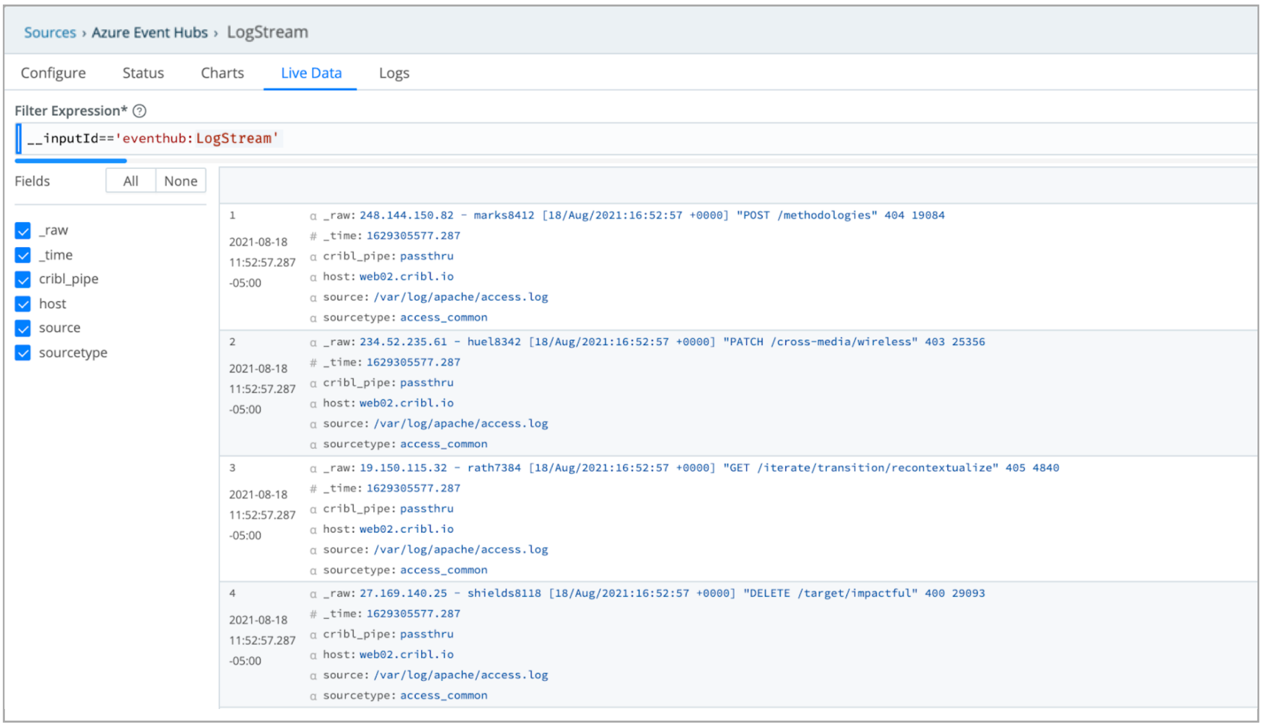
Host ↓	GUID	Status
ip-[redacted]	[redacted]	☑

☑ This source is working properly.

Last refresh: 2021-08-18 16:49:01 UTC

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.



Filter Expression*

```
__inputId=='eventhub:LogStream'
```

Fields

☑ _raw
☑ _time
☑ cribl_pipe
☑ host
☑ source
☑ sourcetype

```
1                 α _raw: 248.144.150.82 - marks8412 [18/Aug/2021:16:52:57 +0000] "POST /methodologies" 404 19084
2021-08-18        # _time: 1629305577.287
11:52:57.287      α cribl_pipe: passthru
-05:00            α host: web02.cribl.io
                  α source: /var/log/apache/access.log
                  α sourcetype: access_common
2                 α _raw: 234.52.235.61 - huel8342 [18/Aug/2021:16:52:57 +0000] "PATCH /cross-media/wireless" 403 25356
2021-08-18        # _time: 1629305577.287
11:52:57.287      α cribl_pipe: passthru
-05:00            α host: web02.cribl.io
                  α source: /var/log/apache/access.log
                  α sourcetype: access_common
3                 α _raw: 19.150.115.32 - rath7384 [18/Aug/2021:16:52:57 +0000] "GET /iterate/transition/recontextualize" 405 4840
2021-08-18        # _time: 1629305577.287
11:52:57.287      α cribl_pipe: passthru
-05:00            α host: web02.cribl.io
                  α source: /var/log/apache/access.log
                  α sourcetype: access_common
4                 α _raw: 27.169.140.25 - shields8118 [18/Aug/2021:16:52:57 +0000] "DELETE /target/impactful" 400 29093
2021-08-18        # _time: 1629305577.287
11:52:57.287      α cribl_pipe: passthru
-05:00            α host: web02.cribl.io
                  α source: /var/log/apache/access.log
                  α sourcetype: access_common
```

Viewing Live Data

;

12.5. 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.

S2S enables TLS compression by default. Do not confuse TLS compression with the `compressed` setting in the Splunk `inputs.conf` file, which is a different thing, and is for non-TLS connections only.

See the Splunk documentation about the `compressed` setting, and about TLS, which Splunk configuration files still refer to as SSL.

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`
- Paid Version in AWS: `https://http-inputs-<cloud_stack_name>:443`
- Paid version in GCP: `https://http-inputs.<cloud_stack_name>:443`

A HEC endpoint for a paid version of Splunk Cloud on AWS, for a company called "Acme Group," might look like this:

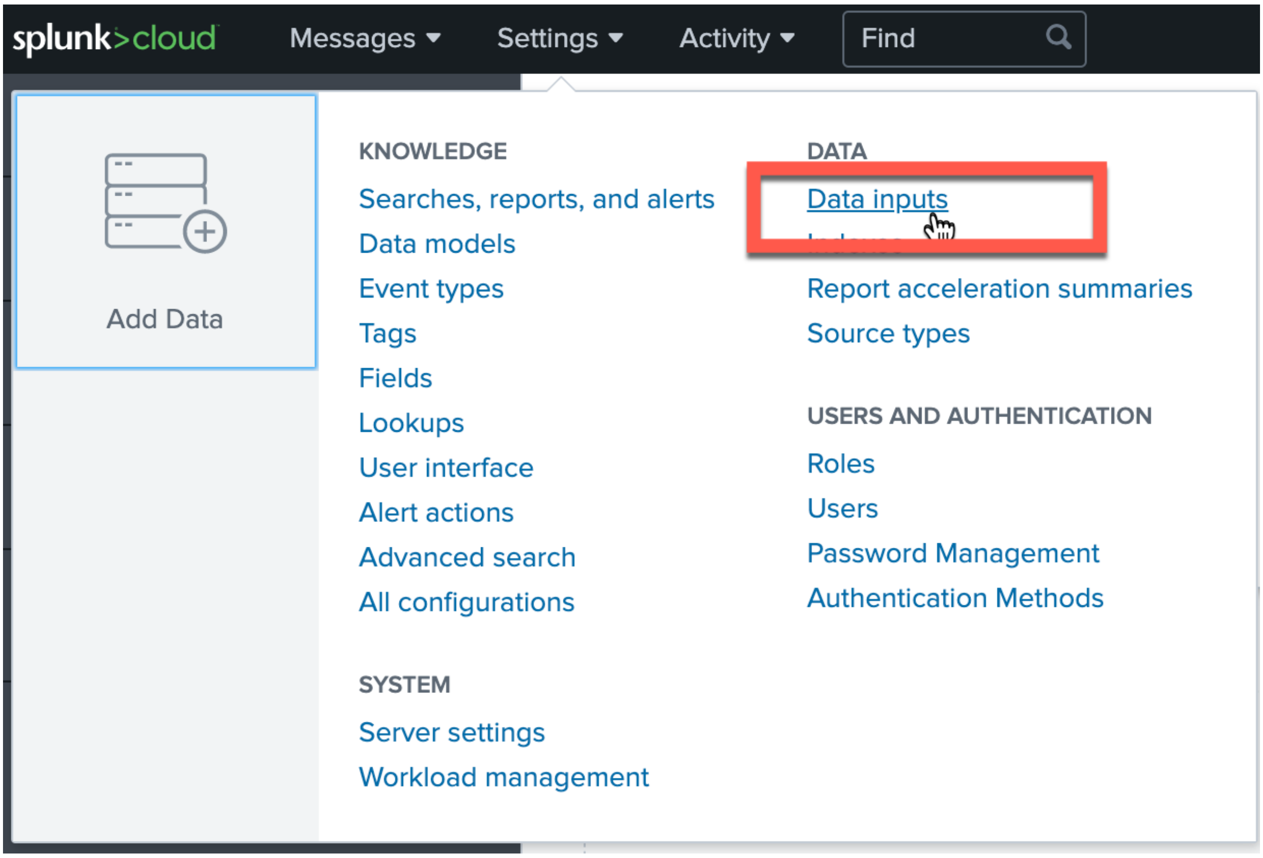
```
https://http-inputs-acmegroup.splunkcloud.com:443
```

Copy the endpoint URL for use when configuring Cribl Stream in the next section.

Create HEC Tokens

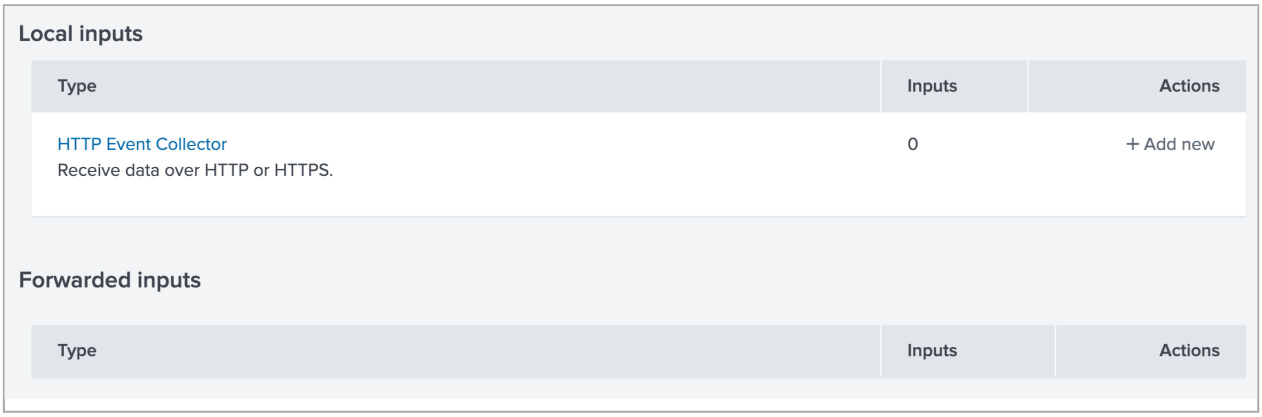
You need to create at least one HEC token. For deployments where you set up routing to individual indexes, or you use HEC tokens for RBAC on Splunk, you will create multiple HEC tokens.

1. In the Splunk UI, open the **Settings** menu and click **Data Inputs**.



Settings > Data inputs

2. In the **HTTP Event Collector** section of the resulting modal, click **+ Add new**.



Local inputs

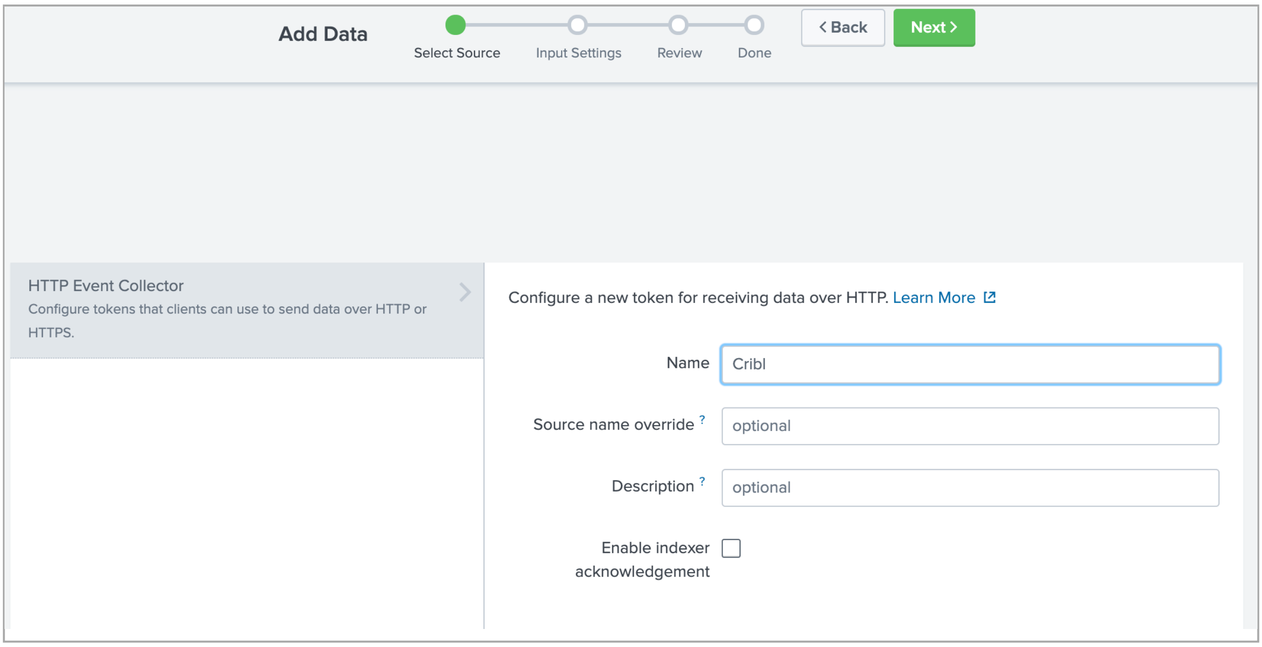
Type	Inputs	Actions
HTTP Event Collector Receive data over HTTP or HTTPS.	0	+ Add new

Forwarded inputs

Type	Inputs	Actions

Adding a new HTTP Event Collector

3. Name the new token and click **Next**.



HTTP Event Collector
Configure tokens that clients can use to send data over HTTP or HTTPS.

Configure a new token for receiving data over HTTP. Learn More

Name: Cribl

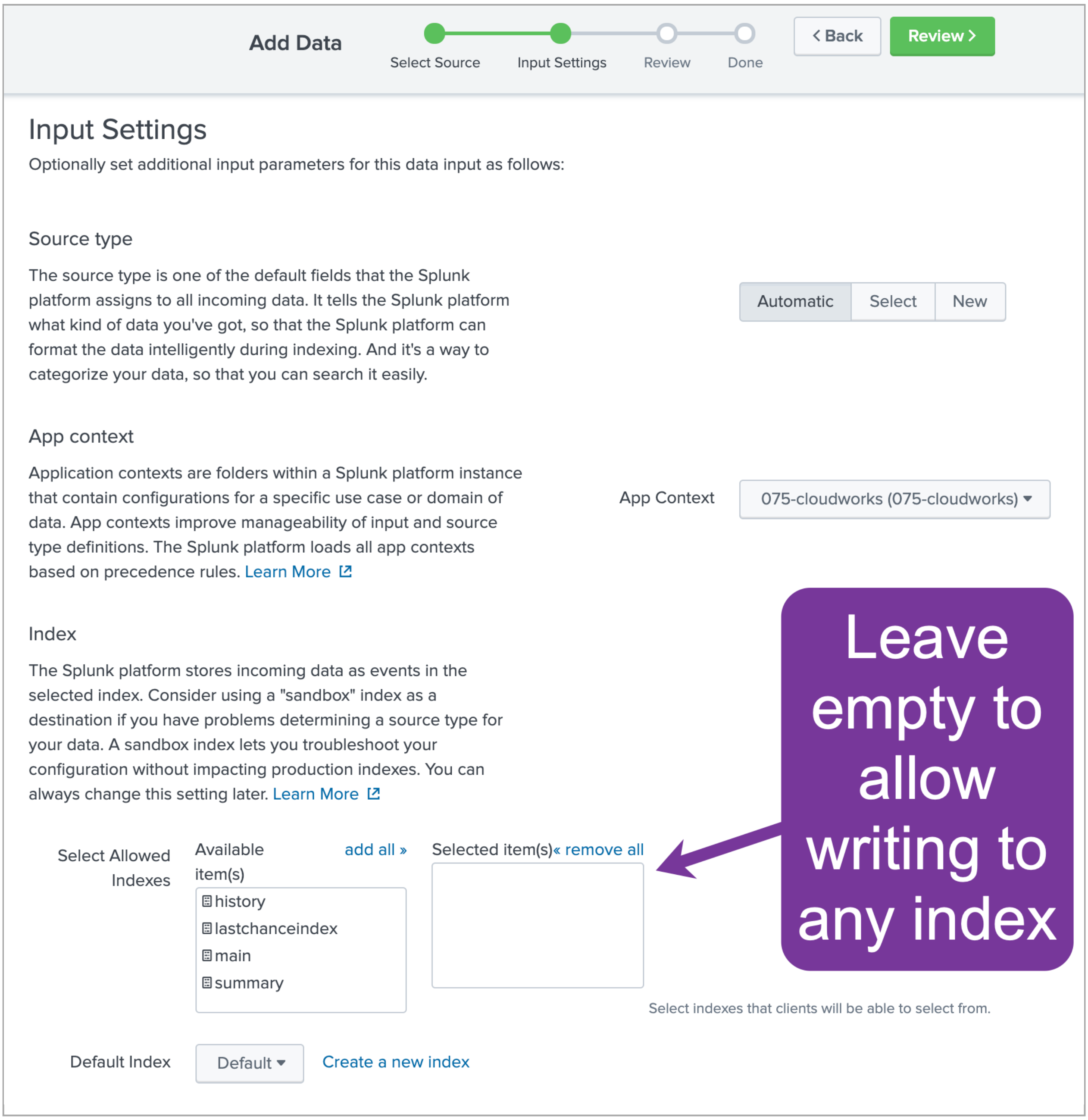
Source name override ?: optional

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.



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.

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

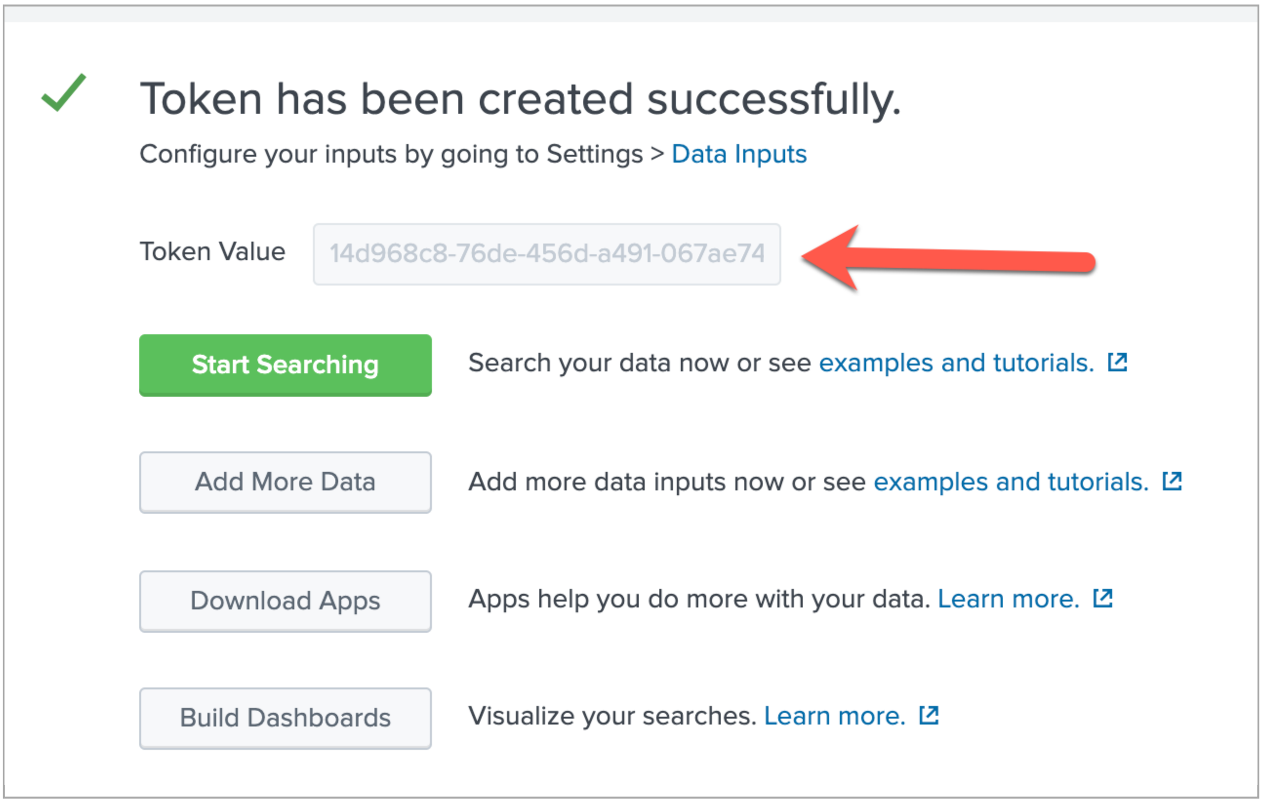
Leave empty to allow writing to any index

Select indexes that clients will be able to select from.

Default Index: Default

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

Search your data now or see examples and tutorials.

Add more data inputs now or see examples and tutorials.

Apps help you do more with your data. Learn more.

Visualize your searches. Learn more.

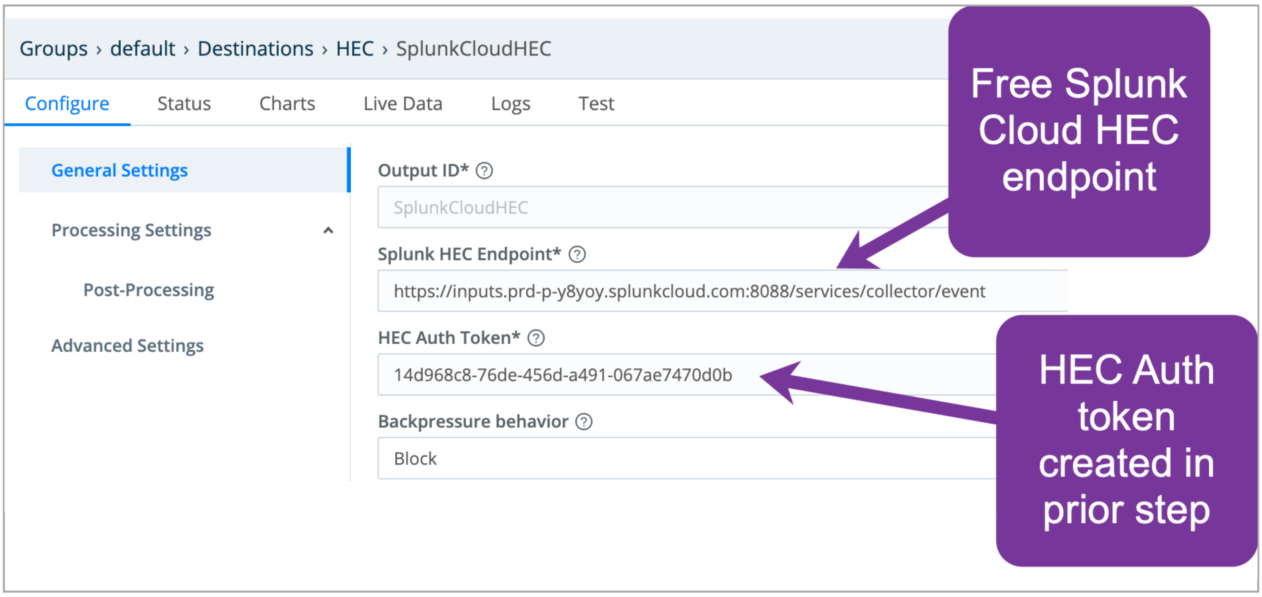
Copying the token value

Add a Splunk HEC Destination in Cribl Stream

From the top nav of a Cribl Stream instance or Group, 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

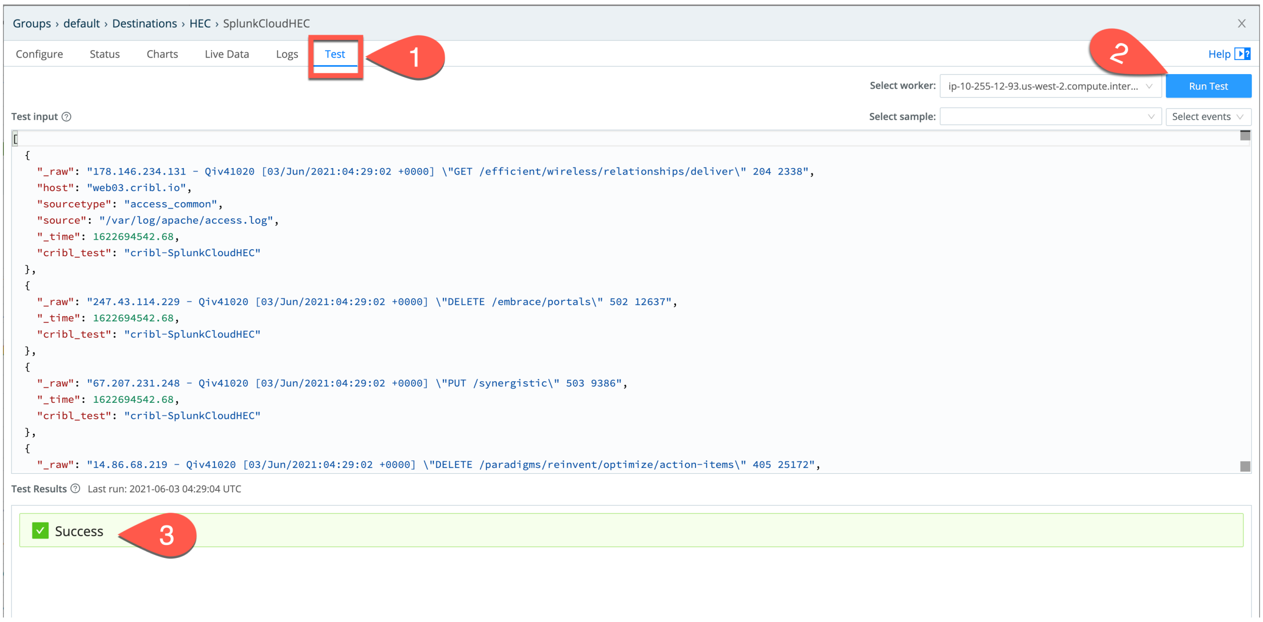
Backpressure behavior: Block

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.



Select worker: ip-10-255-12-93.us-west-2.compute.inter...

Select sample:

Test input

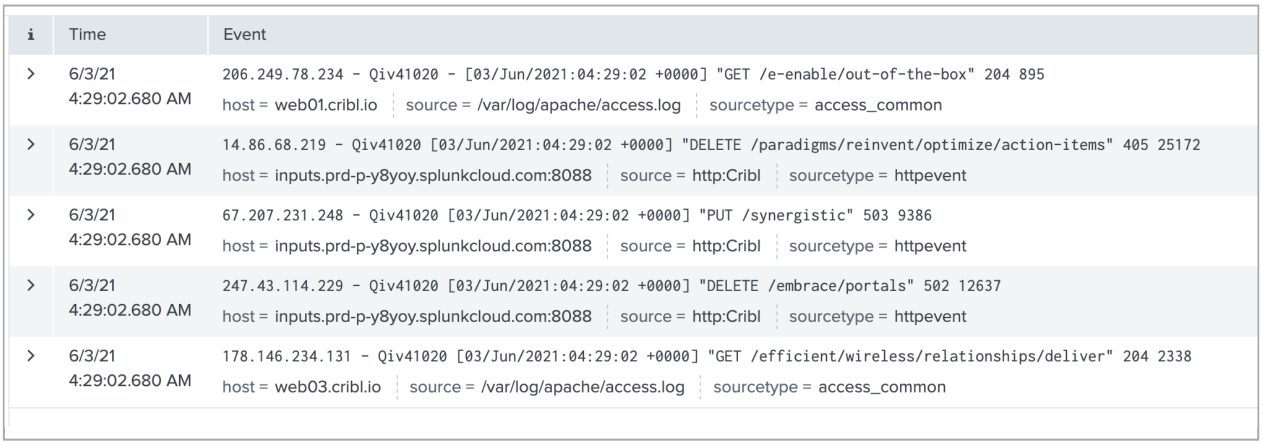
```
[
  {
    "_raw": "178.146.234.131 - Qiv41020 [03/Jun/2021:04:29:02 +0000] \"GET /efficient/wireless/relationships/deliver\" 204 2338",
    "host": "web03.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
    "_time": 1622694542.68,
    "cribl_test": "cribl-SplunkCloudHEC"
  },
  {
    "_raw": "247.43.114.229 - Qiv41020 [03/Jun/2021:04:29:02 +0000] \"DELETE /embrace/portals\" 502 12637",
    "_time": 1622694542.68,
    "cribl_test": "cribl-SplunkCloudHEC"
  },
  {
    "_raw": "67.207.231.248 - Qiv41020 [03/Jun/2021:04:29:02 +0000] \"PUT /synergistic\" 503 9386",
    "_time": 1622694542.68,
    "cribl_test": "cribl-SplunkCloudHEC"
  },
  {
    "_raw": "14.86.68.219 - Qiv41020 [03/Jun/2021:04:29:02 +0000] \"DELETE /paradigms/reinvent/optimize/action-items\" 405 25172",
```

Test Results Last run: 2021-06-03 04:29:04 UTC

✓ Success

Testing the HEC Destination

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

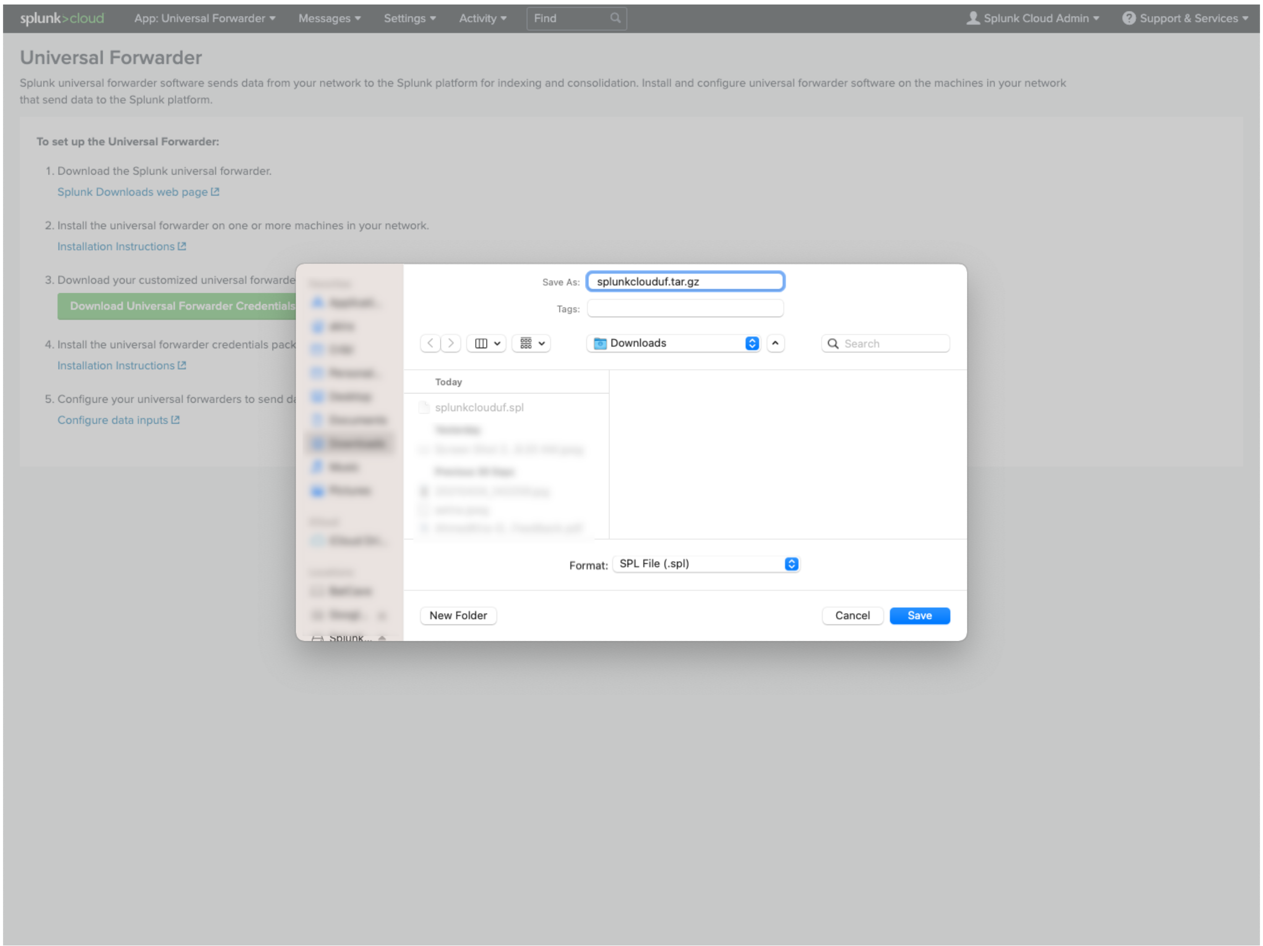
Events flowing to Splunk

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
   Download Universal Forwarder Credentials
4. Install the universal forwarder credentials pack
   Installation Instructions
5. Configure your universal forwarders to send d
   Configure data inputs

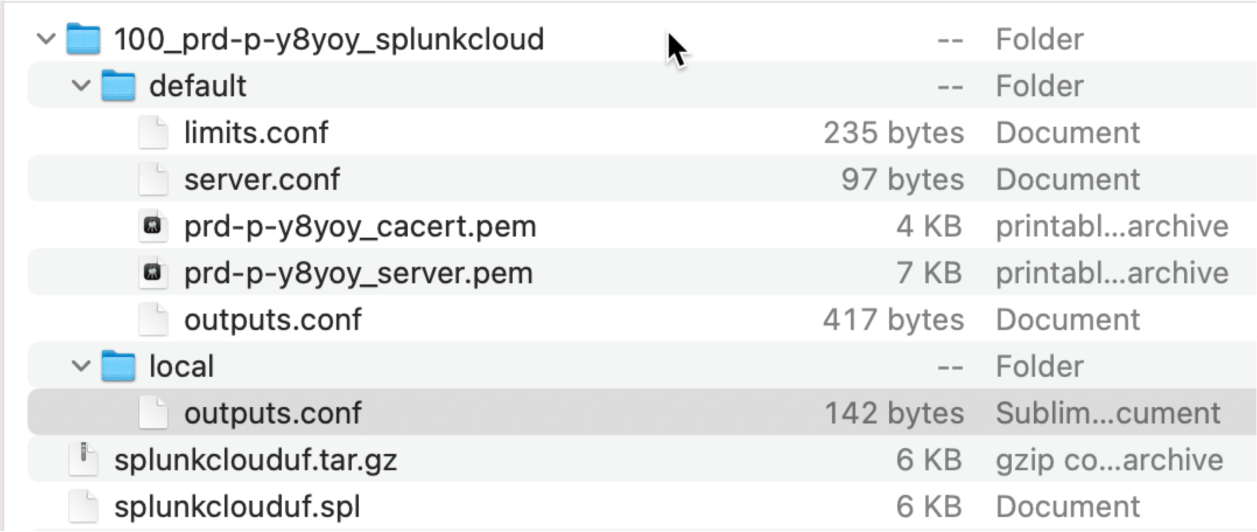
Save As: splunkclouduf.tar.gz

Tags:

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

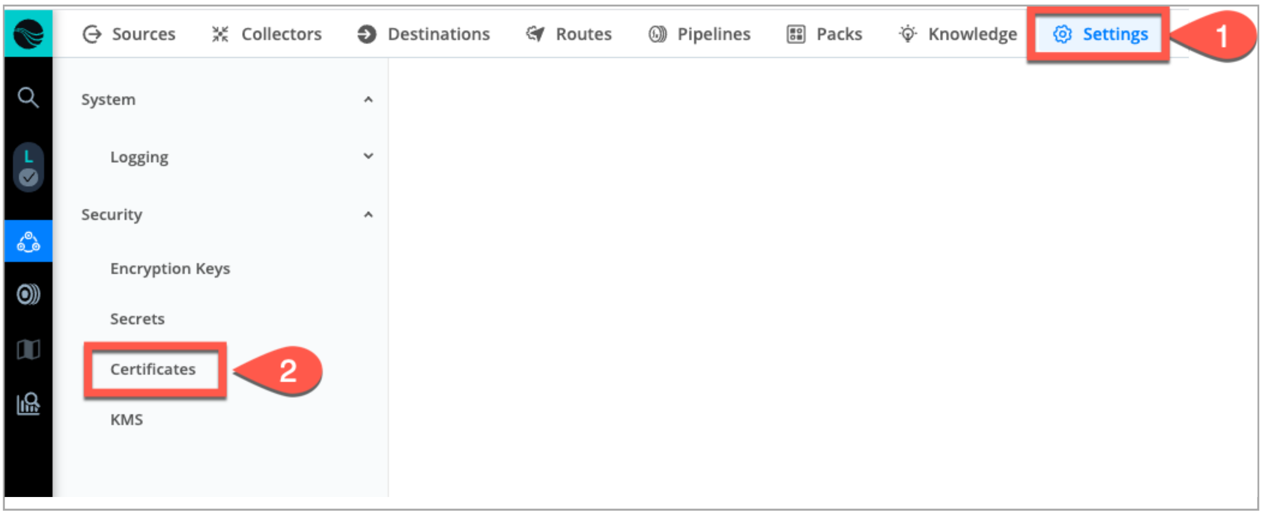
Credentials app files

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

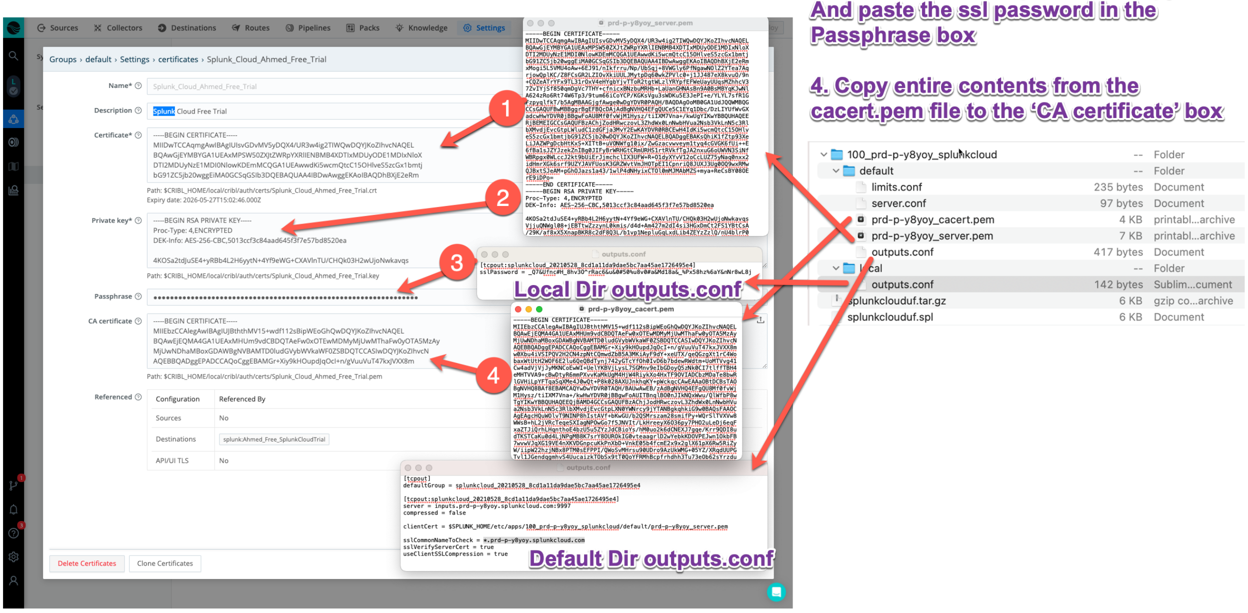
1. In Cribl Stream, select **Groups** > `<group-name>` (or **Configure** > `default`) from the left nav. Then, at the upper right, select **Settings** > **Certificates**.



The Settings > Certificates submenu

2. 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.



1. Copy entire contents from server.pem to the ‘certificate’ box

2. From within the server.pem, locate the private key section and paste it in the ‘private key’ box

3. From the local/outputs.conf, copy And paste the ssl password in the Passphrase box

4. Copy entire contents from the cacert.pem file to the ‘CA certificate’ box

Groups › default › Settings › certificates › Splunk_Cloud_Ahmed_Free_Trial

Name* Splunk_Cloud_Ahmed_Free_Trial

Description Splunk Cloud Free Trial

Certificate*

```
-----BEGIN CERTIFICATE-----
MIIDwTCCAqmgAwIBAgIUIsvGDvMV5yDQX4/UR3w4ig2TIWQwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAxMPSW50ZXJtZWRpYXRlIENBMB4XDTIxMDUyODE1MDIxNloX
DTI2MDUyNzE1MDI0NlowKDEmMCQGA1UEAwwdKi5wcmQtcC15OHlveS5zcGx1bmtj
bG91ZC5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDhBXjE2eRm
```

Path: $CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.crt
Expiry date: 2026-05-27T15:02:46.000Z

Private key*

```
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,5013ccf3c84aad645f3f7e57bd8520ea

4KOSa2tdJuSE4+yRBb4L2H6yytN+4Yf9eWG+CXAVlnTU/CHQk03H2wUjoNwkavqs
```

Path: $CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.key

Passphrase ••

CA certificate

```
-----BEGIN CERTIFICATE-----
MIIEbzCCAlegAwIBAgIUJBththMV15+wdf112sBipWEoGhQwDQYJKoZIhvcNAQEL
BQAwEjEQMA4GA1UEAxMHUm9vdCBDQTAeFw0xOTEwMDMyMjUwMThaFw0yOTA5MzAy
MjUwNDhaMBoxGDAWBgNVBAMTD0ludGVybWVkaWF0ZSBDQTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAMGr+Xiy9kHOupdJqOcI+n/gVuuVuT47kxJVXX8m
```

Path: $CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.pem

Referenced

Configuration	Referenced By
Sources	No
Destinations	splunk:Ahmed_Free_SplunkCloudTrial
API/UI TLS	No

prd-p-y8yoy_server.pem

```
-----BEGIN CERTIFICATE-----
MIIDwTCCAqmgAwIBAgIUIsvGDvMV5yDQX4/UR3w4ig2TIWQwDQYJKoZIhvcNAQEL
BQAwGjEYMBYGA1UEAxMPSW50ZXJtZWRpYXRlIENBMB4XDTIxMDUyODE1MDIxNloX
DTI2MDUyNzE1MDI0NlowKDEmMCQGA1UEAwwdKi5wcmQtcC15OHlveS5zcGx1bmtj
bG91ZC5jb20wggEiMA0GCSqGSIb3DQEBAQUAA4IBDwAwggEKAoIBAQDhBXjE2eRm
xMogi5L5VMU4oAw+6EJ91/nIkfrru/Np/UbSqj+8VWGly6PfNgawNOlZ2YTea7Aq
rjowQplKC/Z8FCsGR2LZIOvXkiUULJMytpDq60wkZPVlc0+j1JJ487eX8kvuO/9n
+CQZeATrYFx9TL31rOxV4eHYgbYjvTToR2tgtWLzlYRYpfEFWeUayUUqsMZhhcV3
7ZvIYjSf850qmDgVc7THY+cfnicxBNzbuMRHb+LaUanGHNAsBn9A0BsMBYqKJwNl
A624zRo6Rt74W6Tp3/9tum66iCoYCP/KGKsVgu3sWDKu5E3JePI+e/YLYL7sfR1G
FzpyqlfkT/b5AgMBAAGjgfAwge0wDgYDVR0PAQH/BAQDAgOoMB0GA1UdJQQWMBQG
CCsGAQUFBwMBBggrBgEFBQcDAjAdBgNVHQ4EFgQUCe5C1EYq1Dbc/DzLIYUfWvGX
adcwHwYDVR0jBBgwFoAU8Mf0fvWjM1Hysz/tiIXM7Vna+/kwUgYIKwYBBQUHAQEE
RjBEMEIGCCsGAQUFBzAChjZodHRwczovL3ZhdWx0LnNwbHVua2Nsb3VkLnN5c3Rl
bXMvdjEvcGtpLWludC1zdGFja3MvY2EwKAYDVR0RBCEwH4IdKi5wcmQtcC15OHlv
eS5zcGx1bmtjbG91ZC5jb20wDQYJKoZIhvcNAQELBQADggEBAKsQhiK1fZtp93Xe
LiJAZWPgDcbHtKxS+XITtB+uVONWfg10ix/ZwGzacvwveym1tyq4cGVGK6fUi++E
6fBa1sJZYJzekZnIBg0JIFyBrWRHGtCRmURHS1rtRVkfTgJA2nxuG6oUWVN3SiNf
WBRpgx0WLccJ2kt9bUiErJjmchclIX3UFW+R+O1dyXYvV12oCcLUZ75yNaq0nxx2
idHmrXGk6srf9UZYJAVFUosK3GRZWvtVmJHOTpEI1CpnriQ8JUXJ3Ug0OQ9wxRMw
QJBxtSJeAM+pGhOJazs1a43/1wlP4dNHyixCTOl0mMJMAbMZS+mya+ReCsBY08OE
rE9iDPo=
-----END CERTIFICATE-----
-----BEGIN RSA PRIVATE KEY-----
Proc-Type: 4,ENCRYPTED
DEK-Info: AES-256-CBC,5013ccf3c84aad645f3f7e57bd8520ea

4KOSa2tdJuSE4+yRBb4L2H6yytN+4Yf9eWG+CXAVlnTU/CHQk03H2wUjoNwkavqs
VjjuQNWgl08+jEBTtwZzzynL0kmis/d4d+Am427m2dI4si3HGxDmCt2FS1YBtCsA
/29K/af8xX5XnapBKR8c2dF8Q3L/b1vp1NepluGqLxdLib4ZEYzZzlQ/nU4blrP0
```

outputs.conf

```
[tcpout:splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4]
sslPassword = _Q7&Ufnc#H_8hv30^rRac6&u&0#50%u8v0#a&Md18a&_%Px58hz%6aY&nNr8wL8j
```

prd-p-y8yoy_cacert.pem

```
-----BEGIN CERTIFICATE-----
MIIEbzCCAlegAwIBAgIUJBththMV15+wdf112sBipWEoGhQwDQYJKoZIhvcNAQEL
BQAwEjEQMA4GA1UEAxMHUm9vdCBDQTAeFw0xOTEwMDMyMjUwMThaFw0yOTA5MzAy
MjUwNDhaMBoxGDAWBgNVBAMTD0ludGVybWVkaWF0ZSBDQTCCASIwDQYJKoZIhvcN
AQEBBQADggEPADCCAQoCggEBAMGr+Xiy9kHOupdJqOcI+n/gVuuVuT47kxJVXX8m
w0Xbu4iVSIPQV2H2CN4zpNtCQmwdZbB5A3MKiAyF9dY+xeUTX/qeQGzqXt1rC4Wo
baxWtUtH2WOF6E2lu6QeQBdTynj742yGTcYfOh0IvD6b7bdewRWdtm+UoMTVvg41
Cw4adVjVjJyMKNCoEwWI+UelYKBVjLysL7SGMnv9eIbGDoyQ5zNk0CI7tlffTBH4
eMHTVVA9+cBwDtyR6mmPXvvKaMkUgM4HjW4RiykXo4HxTF9OVIADCbzMDaTe8bwR
lGVHiLpYFTqaSqXMe4J0wQt+P8k028AXUJnkhqKY+pWckqcCAwEAAaOBtDCBsTAO
BgNVHQ8BAf8EBAMCAQYwDwYDVR0TAQH/BAUwAwEB/zAdBgNVHQ4EFgQU8Mf0fvWj
M1Hysz/tiIXM7Vna+/kwHwYDVR0jBBgwFoAUITBnqlBO0nJIkNQxWwu/QlWfbP8w
TgYIKwYBBQUHAQEEQjBAMD4GCCsGAQUFBzAChjJodHRwczovL3ZhdWx0LnNwbHVu
a2Nsb3VkLnN5c3RlbXMvdjEvcGtpLXN0YWNrcy9jYTANBgkqhkiG9w0BAQsFAAOC
AgEAgcHQuWOlvT9NINP8hIstAVf+bKwGU/b2QSMrszam28smifPy+WQrSlTVXVw8
WWsB+hL2jVRcTeqeSXIagNPOwGo7f5JNVIt/LkHreeyX6O36py7PHO2uLeDj6eqF
xaZTJiQrhLHqnthoE4bzU5u5ZYzJdCBioYs/hM0uo2k6dCNEXJ7gqe/Krr9QDI8u
dTKSTCaKu0d4LjNPgMB8K7srY8OUROkIG0vteaagrlD2wYebkKDOVPEJwn10kbFB
7wvwVJqXG19VE4nXKVDGnpcuKkPnXbD+VnkE05b4fcmE2x9x2glX61pX6Rw5RiZy
W/iipW22hzjNBx8PTM0sEFPPI/QWoSvMHrsu90UDro9AzUkWMG+05YZ/XRqdUUPG
Tvl1JGendqgmhvS4UucaizkTObSx9tT0QoYFRMhBcpfrhdhh3Tu73e0b62sYrzdu
```

outputs.conf

```
[tcpout]
defaultGroup = splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4

[tcpout:splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4]
server = inputs.prd-p-y8yoy.splunkcloud.com:9997
compressed = false

clientCert = $SPLUNK_HOME/etc/apps/100_prd-p-y8yoy_splunkcloud/default/prd-p-y8yoy_server.pem

sslCommonNameToCheck = *.prd-p-y8yoy.splunkcloud.com
sslVerifyServerCert = true
useClientSSLCompression = true
```

```
∨ 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
```

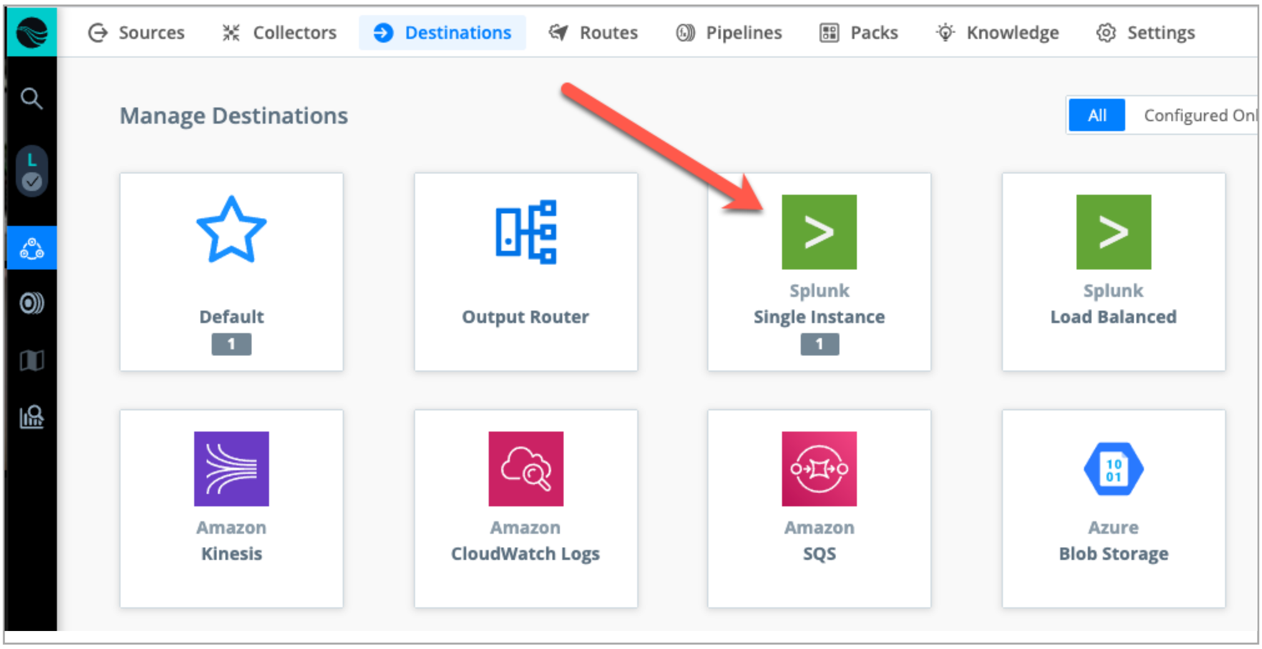
Copying certificate values

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.

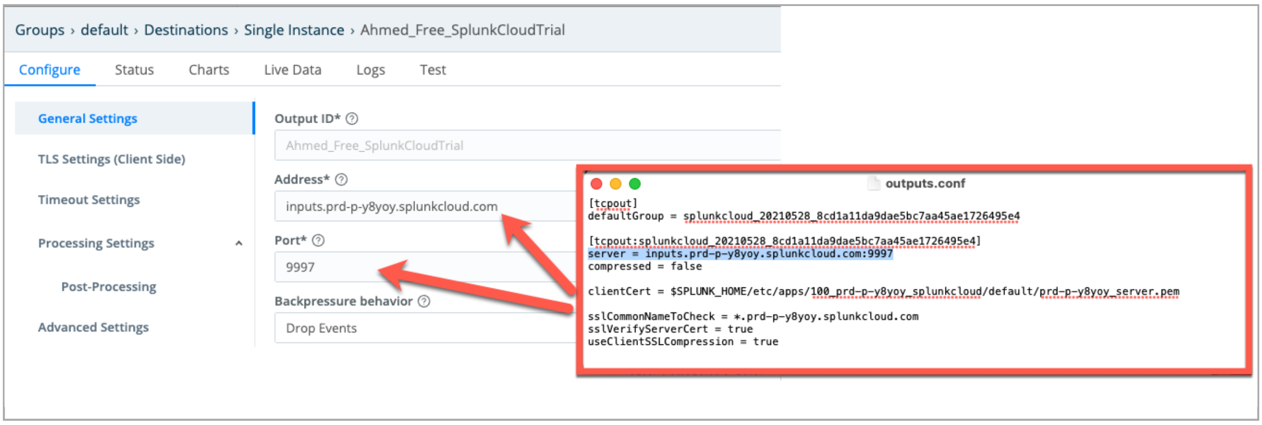
1. From the top nav of a Cribl Stream instance or Group, select **Data** > **Destinations**, then select either **Splunk** > **Load Balanced** or **Splunk** > **Single Instance** from the **Manage Destinations** page's tiles or left nav. Then click **+ Add New** to open the corresponding **New Destination** modal.



Creating the Splunk Destination

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.



Output ID* Ahmed_Free_SplunkCloudTrial

Address* inputs.prd-p-y8yoy.splunkcloud.com

Port* 9997

Backpressure behavior Drop Events

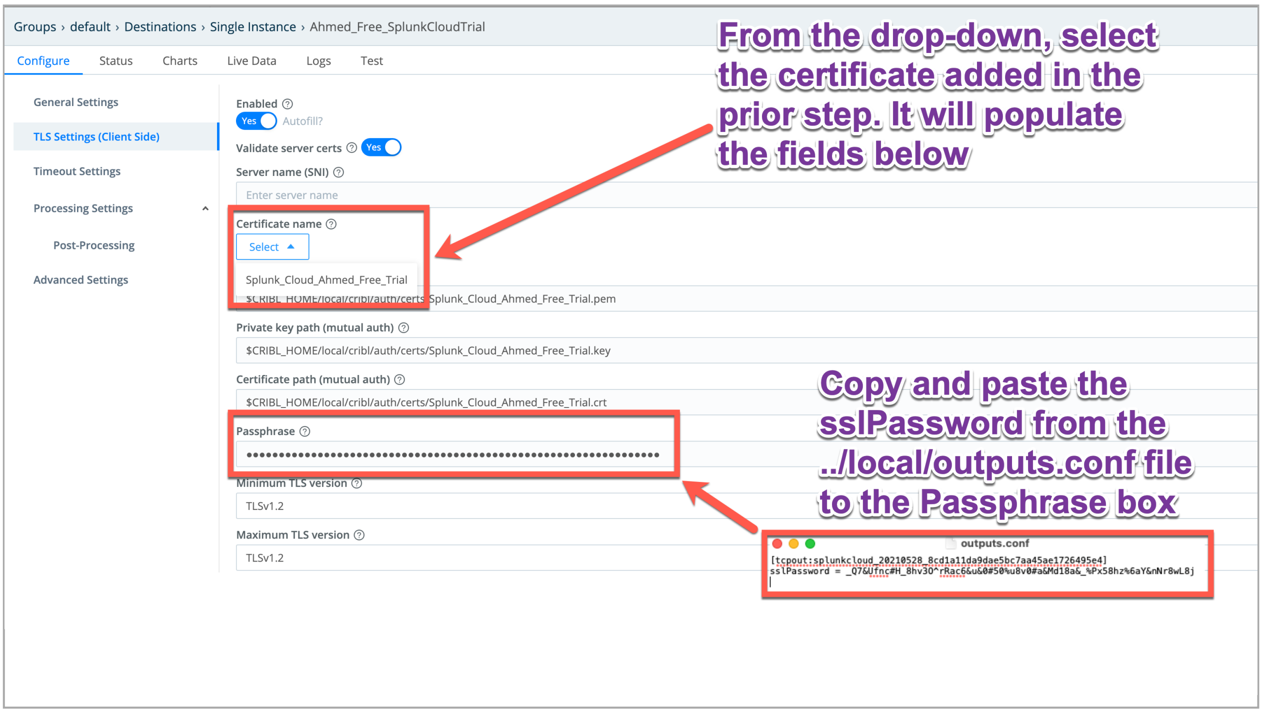
outputs.conf

```
[tcpout]
defaultGroup = splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4

[tcpout:splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4]
server = inputs.prd-p-y8yoy.splunkcloud.com:9997
compressed = false

clientCert = $SPLUNK_HOME/etc/apps/100_prd-p-y8yoy_splunkcloud/default/prd-p-y8yoy_server.pem

sslCommonNameToCheck = *.prd-p-y8yoy.splunkcloud.com
sslVerifyServerCert = true
useClientSSLCompression = true
```

The General Settings tab

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**.



Enabled: Yes

Validate server certs: Yes

Server name (SNI): Enter server name

Certificate name: Splunk_Cloud_Ahmed_Free_Trial

$CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.pem

Private key path (mutual auth): $CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.key

Certificate path (mutual auth): $CRIBL_HOME/local/cribl/auth/certs/Splunk_Cloud_Ahmed_Free_Trial.crt

Passphrase: ••

Minimum TLS version: TLSv1.2

Maximum TLS version: TLSv1.2

From the drop-down, select the certificate added in the prior step. It will populate the fields below

Copy and paste the sslPassword from the ../local/outputs.conf file to the Passphrase box

outputs.conf

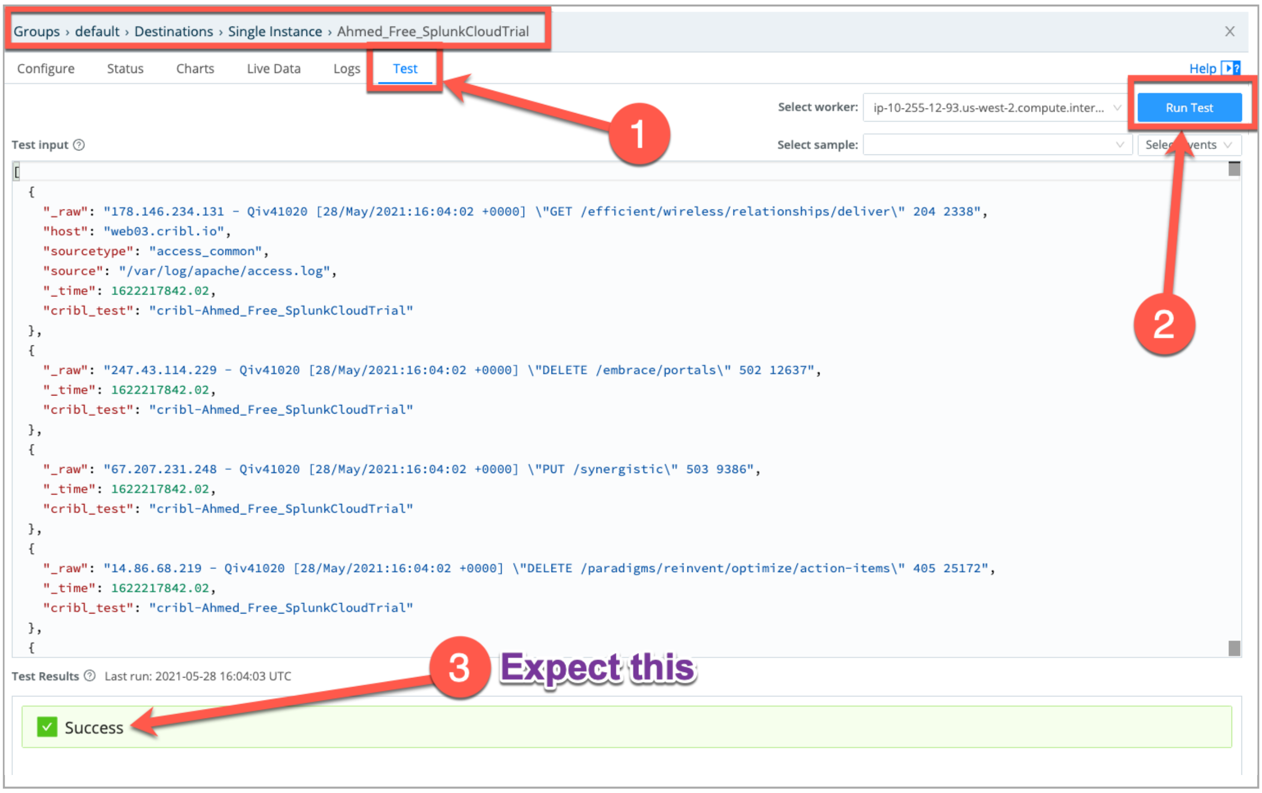
```
[tcpout:splunkcloud_20210528_8cd1a11da9dae5bc7aa45ae1726495e4]
sslPassword = _Q7&Ufnc#H_8hv3O^rRac6&u&0#50%u8v0#a&Md18a&_%Px58hz%6aY&nNr8wL8j
```

TLS Settings

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.

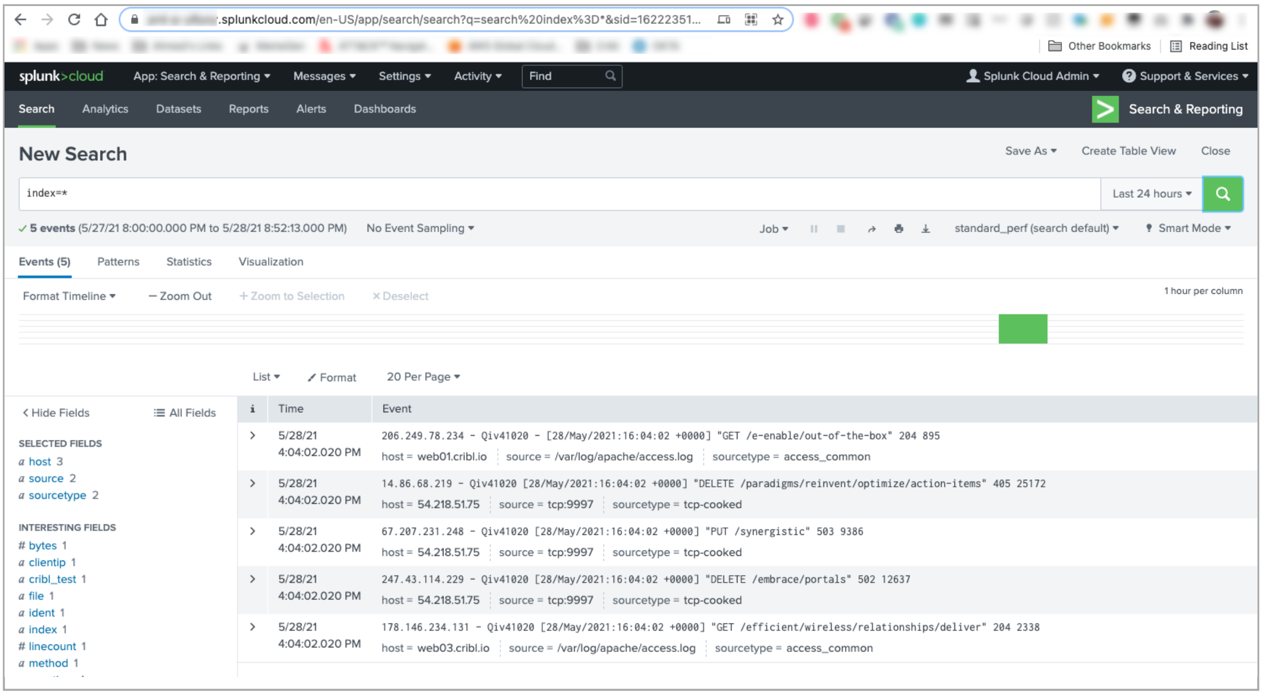
1. In Cribl Stream, open the Destination that you created in the previous section.
   - In the configuration modal's **Test** tab, click **Run Test**.
   - You should see a **Success** message.


Groups › default › Destinations › Single Instance › Ahmed_Free_SplunkCloudTrial
Configure
Status
Charts
Live Data
Logs
Test
Help
Select worker:
ip-10-255-12-93.us-west-2.compute.inter...
Run Test
Select sample:
Select events
Test input
1
2
3
Expect this
Test Results
Last run: 2021-05-28 16:04:03 UTC
Success

```
[
  {
    "_raw": "178.146.234.131 - Qiv41020 [28/May/2021:16:04:02 +0000] \"GET /efficient/wireless/relationships/deliver\" 204 2338",
    "host": "web03.cribl.io",
    "sourcetype": "access_common",
    "source": "/var/log/apache/access.log",
    "_time": 1622217842.02,
    "cribl_test": "cribl-Ahmed_Free_SplunkCloudTrial"
  },
  {
    "_raw": "247.43.114.229 - Qiv41020 [28/May/2021:16:04:02 +0000] \"DELETE /embrace/portals\" 502 12637",
    "_time": 1622217842.02,
    "cribl_test": "cribl-Ahmed_Free_SplunkCloudTrial"
  },
  {
    "_raw": "67.207.231.248 - Qiv41020 [28/May/2021:16:04:02 +0000] \"PUT /synergistic\" 503 9386",
    "_time": 1622217842.02,
    "cribl_test": "cribl-Ahmed_Free_SplunkCloudTrial"
  },
  {
    "_raw": "14.86.68.219 - Qiv41020 [28/May/2021:16:04:02 +0000] \"DELETE /paradigms/reinvent/optimize/action-items\" 405 25172",
    "_time": 1622217842.02,
    "cribl_test": "cribl-Ahmed_Free_SplunkCloudTrial"
  },
  {
```

Testing the Splunk Destination

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.



```
index=*
```

✓ 5 events (5/27/21 8:00:00.000 PM to 5/28/21 8:52:13.000 PM)   No Event Sampling

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.

;

12.6. Configuring WEF for Cribl Stream

In Cribl.Cloud, 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.

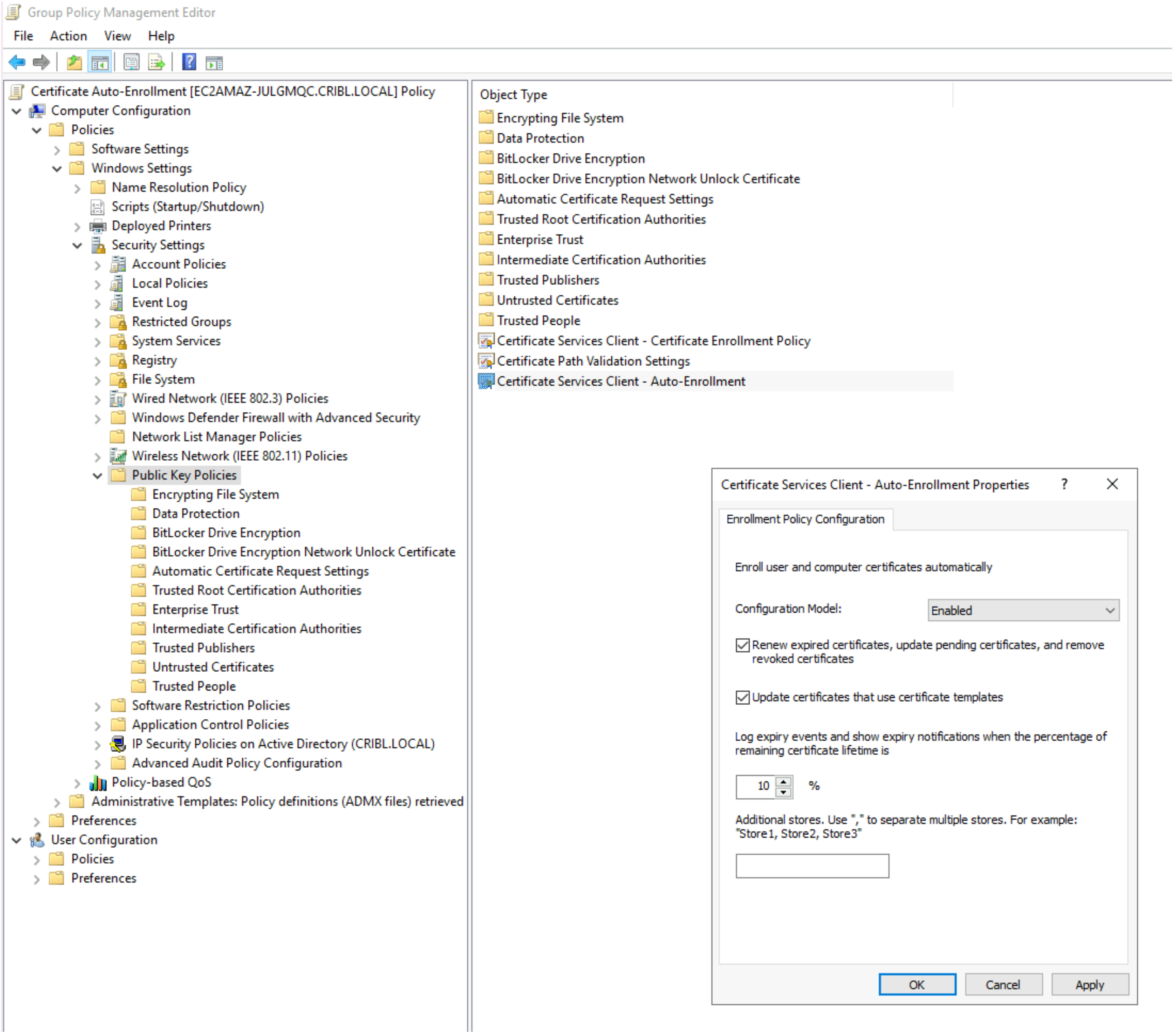
Configuring Certificate Auto-Enrollment

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
OK
Cancel
Apply

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

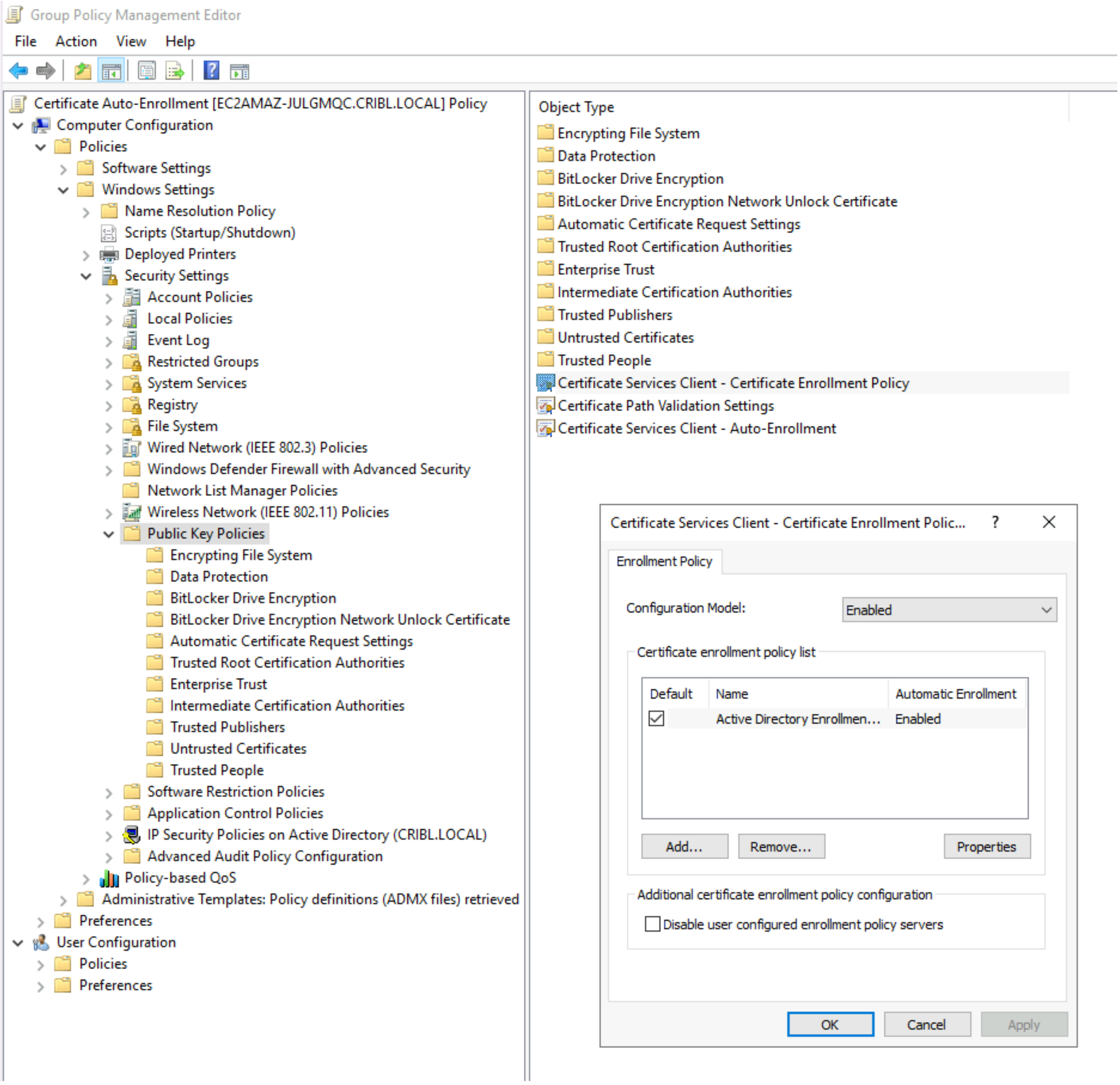
10 %

Additional stores. Use "," to separate multiple stores. For example: "Store1, Store2, Store3"

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
Add...
Remove...
Properties
OK
Cancel
Apply

Configuration Model: Enabled

Certificate enrollment policy list

Default	Name	Automatic Enrollment
☑	Active Directory Enrollmen...	Enabled

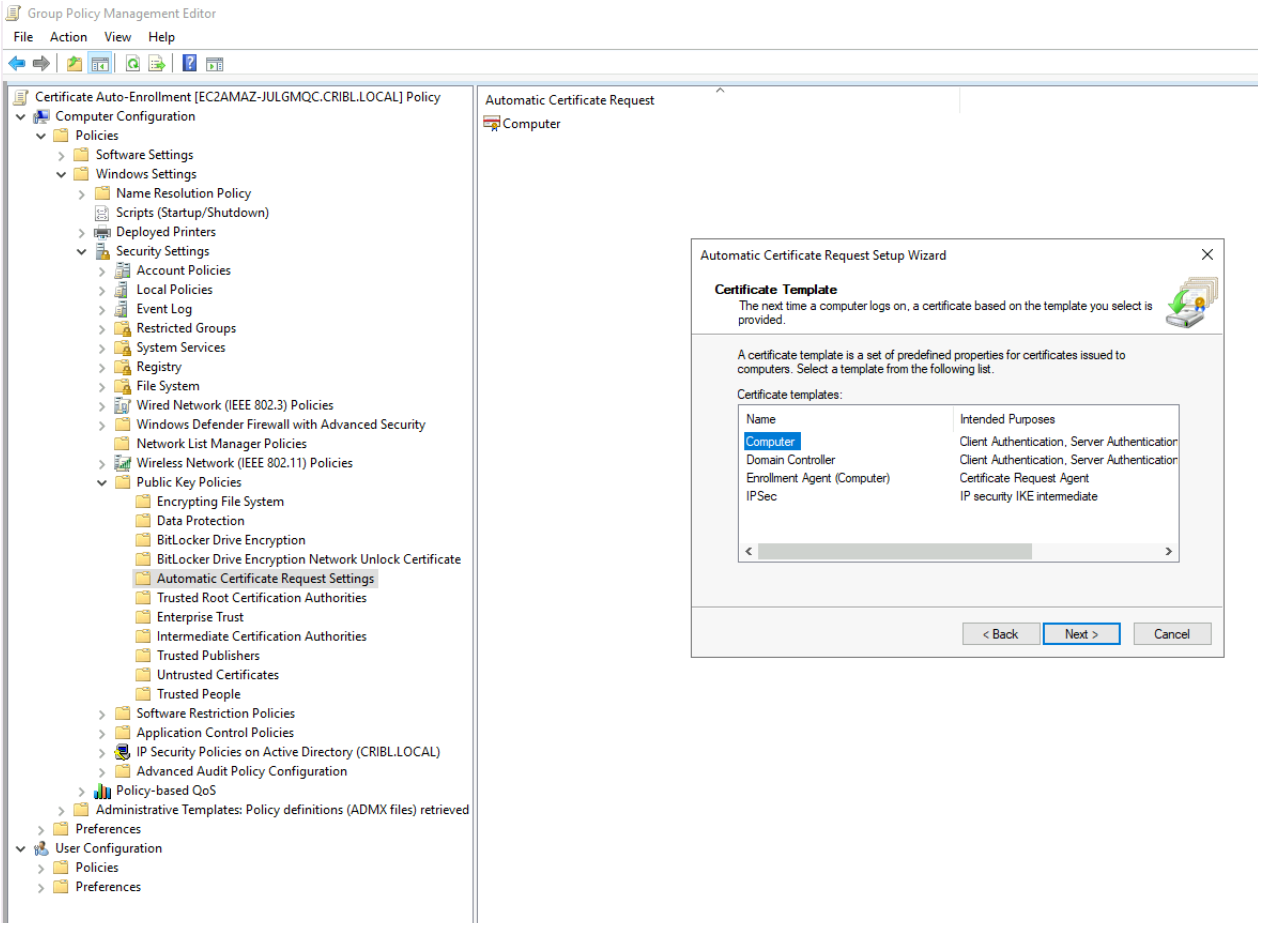
Additional certificate enrollment policy configuration

☐ Disable user configured enrollment policy servers

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 -
DnsName "Cribl Windows Event Forwarding CA" -subject "CN=CRIBL-CA, DC=cribl,
DC=local" -KeyUsage CertSign
Export-Certificate -Cert $rootca -FilePath C:\temp\cribl-wef-ca.cer
Import-Certificate -FilePath C:\temp\cribl-wef-ca.cer -CertStoreLocation
cert:\LocalMachine\Root
```

Now, generate the client certificate.

```
New-SelfSignedCertificate -CertStoreLocation cert:\LocalMachine\My -DnsName cribl-
wef-client -Signer $rootca
```

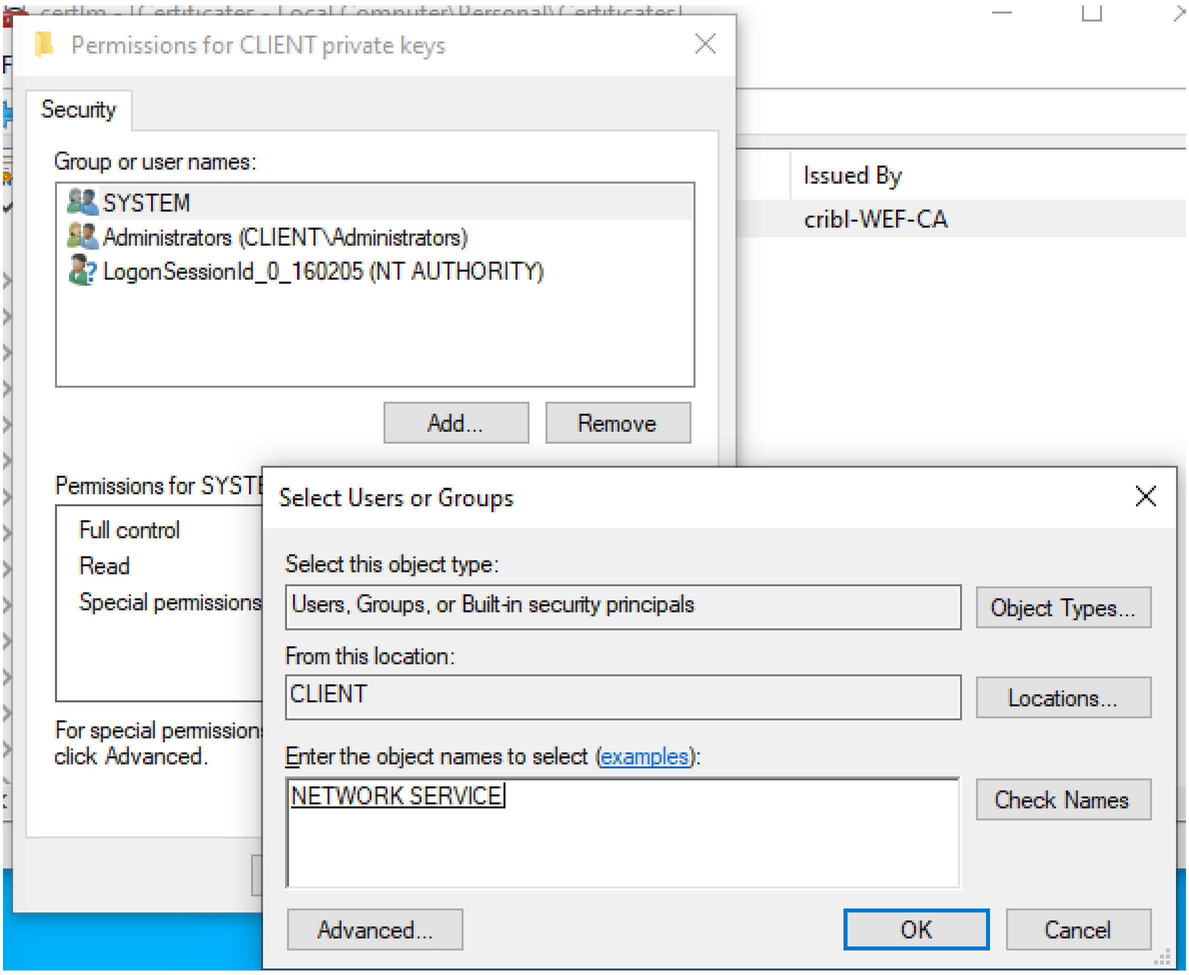
Setting Client Certificate Permissions

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.



Permissions for CLIENT private keys

Group or user names:

SYSTEM
Administrators (CLIENT\Administrators)
LogonSessionId_0_160205 (NT AUTHORITY)

Permissions for SYST

Full control
Read
Special permissions

For special permission
click Advanced.

Select Users or Groups

Select this object type:
Users, Groups, or Built-in security principals

From this location:
CLIENT

Enter the object names to select (examples):
NETWORK SERVICE

Issued By
cribl-WEF-CA

Adding the Network Service user

Cert Permissions via Command Line

As an alternative, you can run the following PowerShell script (as administrator):

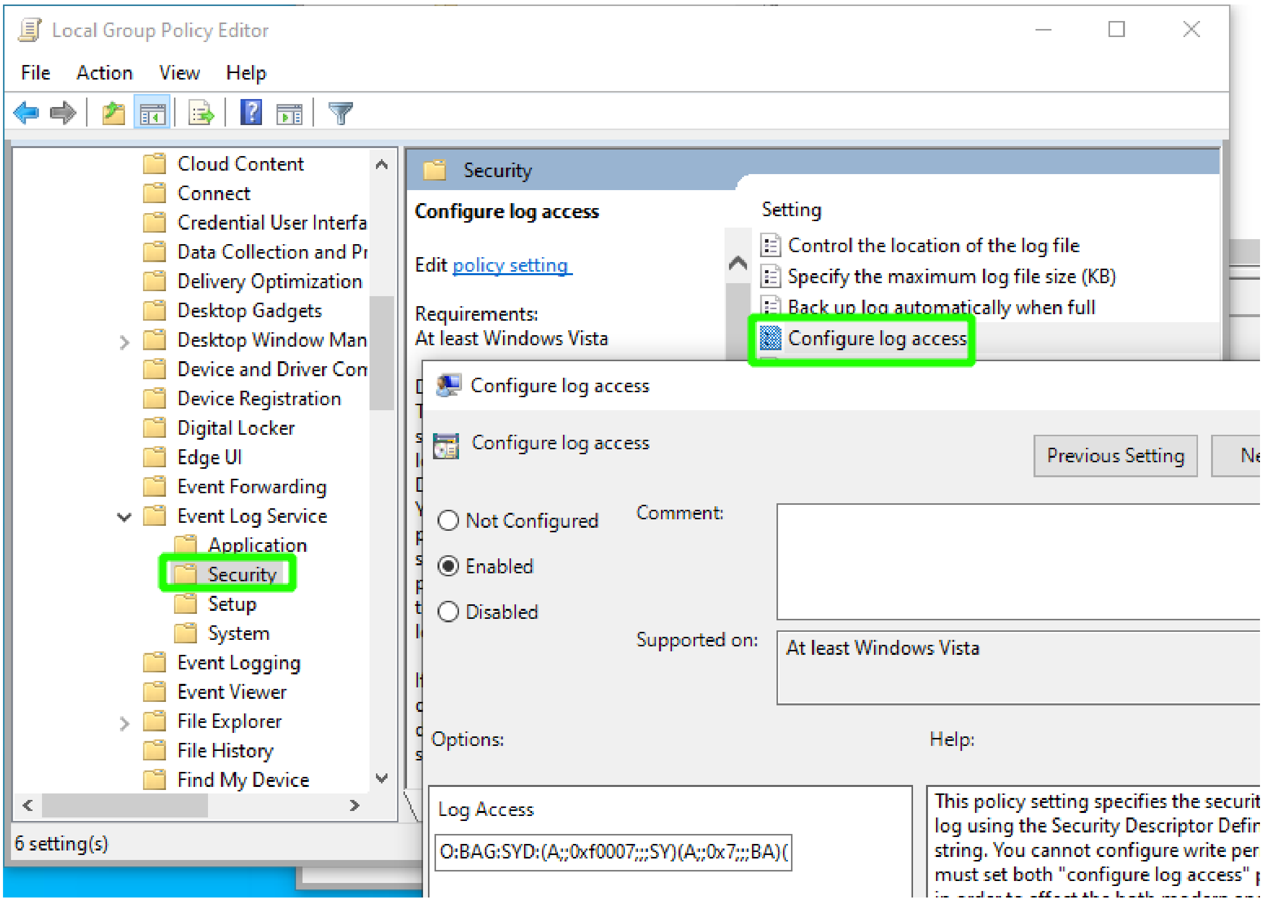
```
$issuer = "CN=CRIBL-CA, DC=cribl, DC=local"
$certs = Get-ChildItem -Path cert:\LocalMachine\My | Where {$_.Issuer -eq $issuer}

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.FileSystemAccessRule -ArgumentList $permission

  # Location of the machine-related keys
  $keyPath = Join-Path -Path $env:ProgramData -ChildPath
"\Microsoft\Crypto\RSA\MachineKeys"
  $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.

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

Supported on: At least Windows Vista

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" p

Configuring log access

In the resulting modal, under **Options** > **Log Access**, enter the following Log Access configuration:

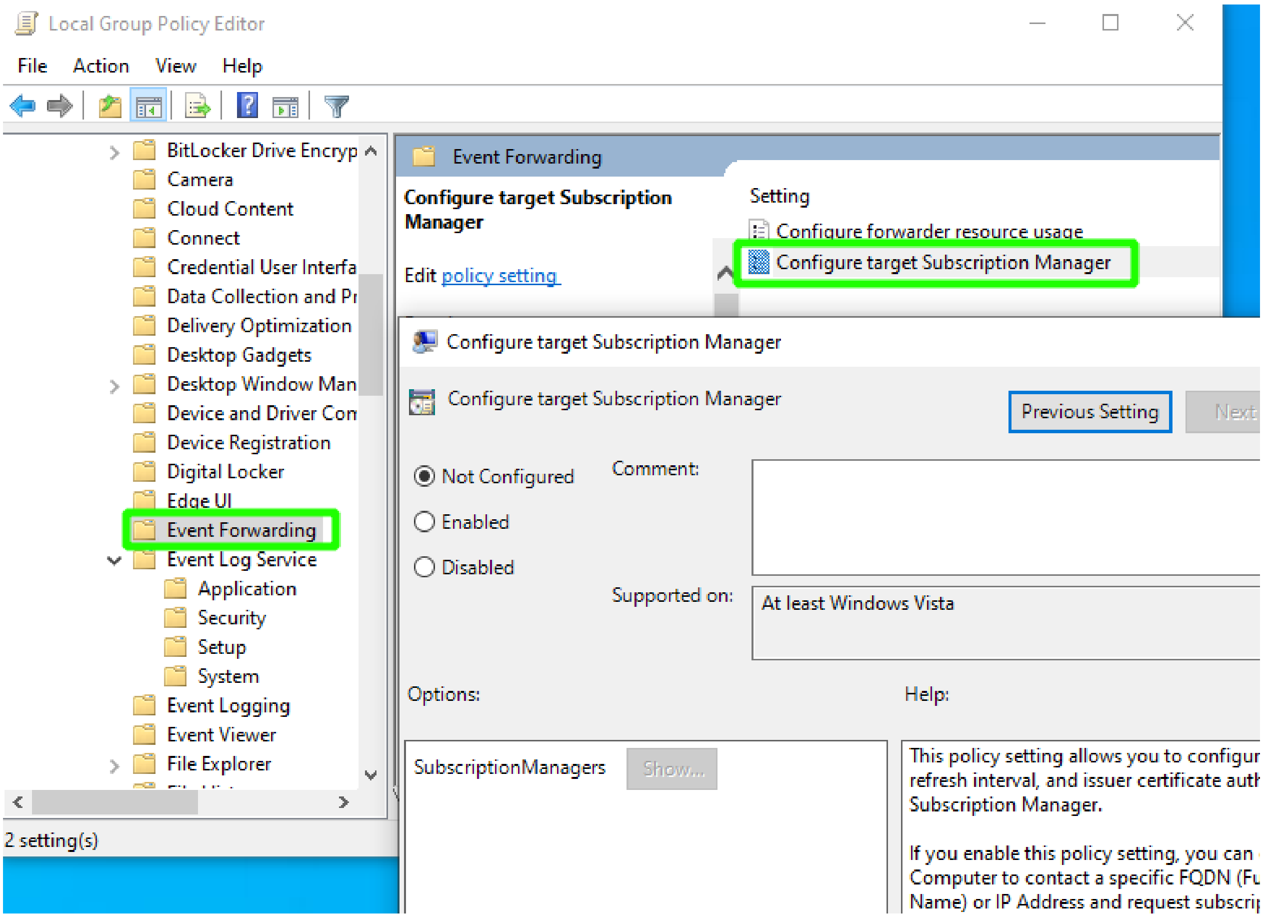
```
O:BAG:SYD:(A;;0xf0007;;;SY)(A;;0x7;;;BA)(A;;0x1;;;BO)(A;;0x1;;;SO)(A;;0x1;;;S-1-5-
32-573)(A;;0x1;;;S-1-5-20)
```

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.



● Not Configured
○ Enabled
○ Disabled

Comment:

Supported on: At least Windows Vista

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.logstream.<organization>.cribl.cloud:<port>/wsman/SubscriptionManager
```

If you forget your Client CA hash, run the following command to find it:

```
Get-ChildItem cert:\LocalMachine\root | Where-Object {$_.Subject -Like
"CN=Cribl*"}
```

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\root | Where-Object {$_.Subject -Like "CN=Cribl*"}).Thumbprint)
$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 that for the `cribl-wef-ca` certificate, you need to generate them to pass validation.

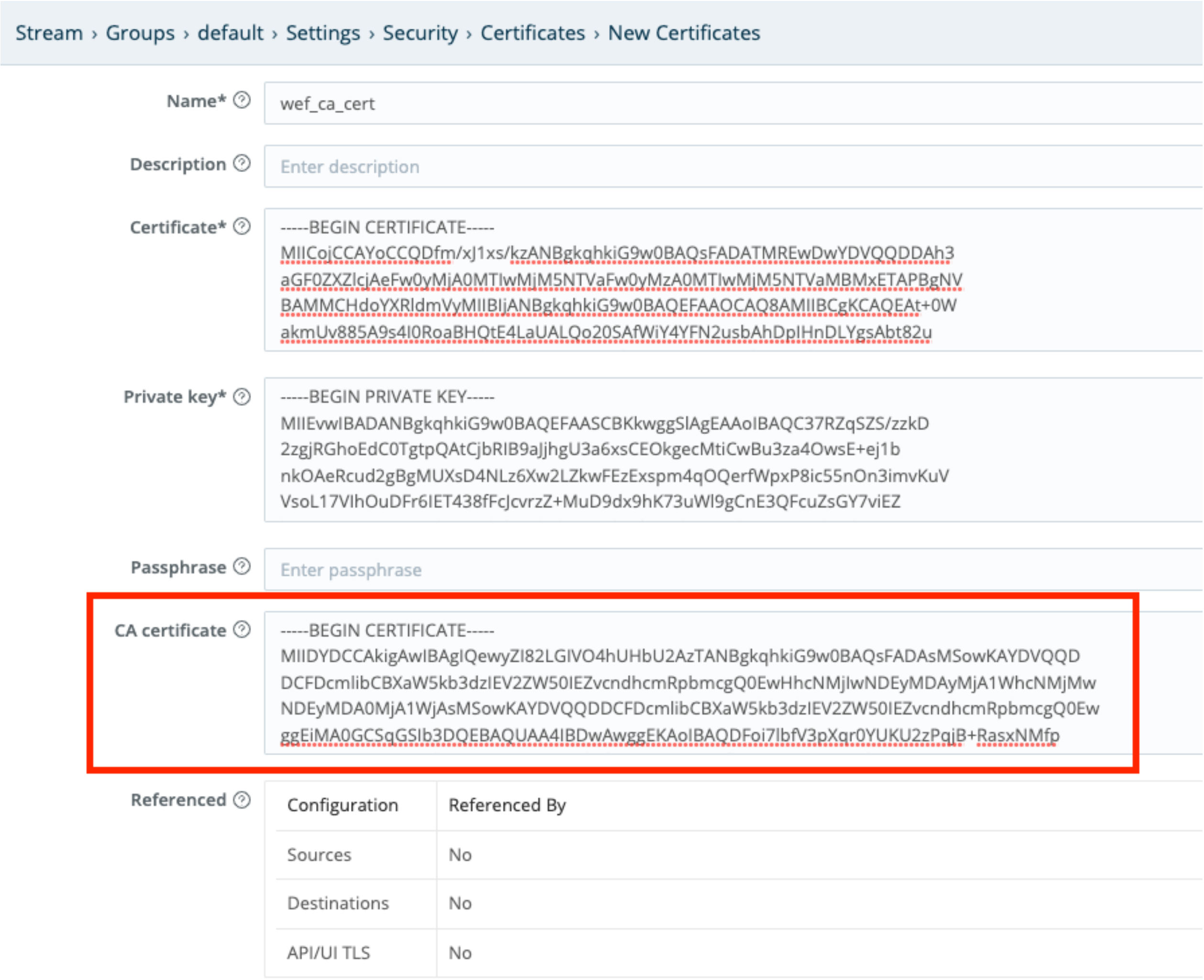
Run the following `openssl` command to generate a placeholder cert/key pair:

```
openssl req -x509 -newkey rsa:2048 -nodes -keyout key.pem -out cert.pem -sha256 -
days 365 -subj '/CN=placeholder'
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 any order) in the **CA certificate** field.



Stream › Groups › default › Settings › Security › Certificates › New Certificates

Name* wef_ca_cert

Description Enter description

Certificate*

```
-----BEGIN CERTIFICATE-----
MIICojCCAYoCCQDfm/xJ1xs/kzANBgkqhkiG9w0BAQsFADATMREwDwYDVQQDDAh3
aGF0ZXZlcjAeFw0yMjA0MTIwMjM5NTVaFw0yMzA0MTIwMjM5NTVaMBMxETAPBgNV
BAMMCHdoYXRldmVyMIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAt+0W
akmUv885A9s4I0RoaBHQtE4LaUALQo20SAfWiY4YFN2usbAhDpIHnDLYgsAbt82u
```

Private key*

```
-----BEGIN PRIVATE KEY-----
MIIEvwIBADANBgkqhkiG9w0BAQEFAASCBKkwggSlAgEAAoIBAQC37RZqSZS/zzkD
2zgjRGhoEdC0TgtpQAtCjbRIB9aJjhgU3a6xsCEOkgecMtiCwBu3za4OwsE+ej1b
nkOAeRcud2gBgMUXsD4NLz6Xw2LZkwFEzExspm4qOQerfWpxP8ic55nOn3imvKuV
VsoL17VIhOuDFr6IET438fFcJcvrzZ+MuD9dx9hK73uWl9gCnE3QFcuZsGY7viEZ
```

Passphrase Enter passphrase

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 certicate 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 **+ New Source**, or click **+ Add** beside **Sources**. From the resulting drawer's tiles, select [**Push** >] **Windows Event Forwarder**. Next, click either **+ Add New** or (if displayed) **Select Existing** to open a Windows Event Forwarder Source drawer.

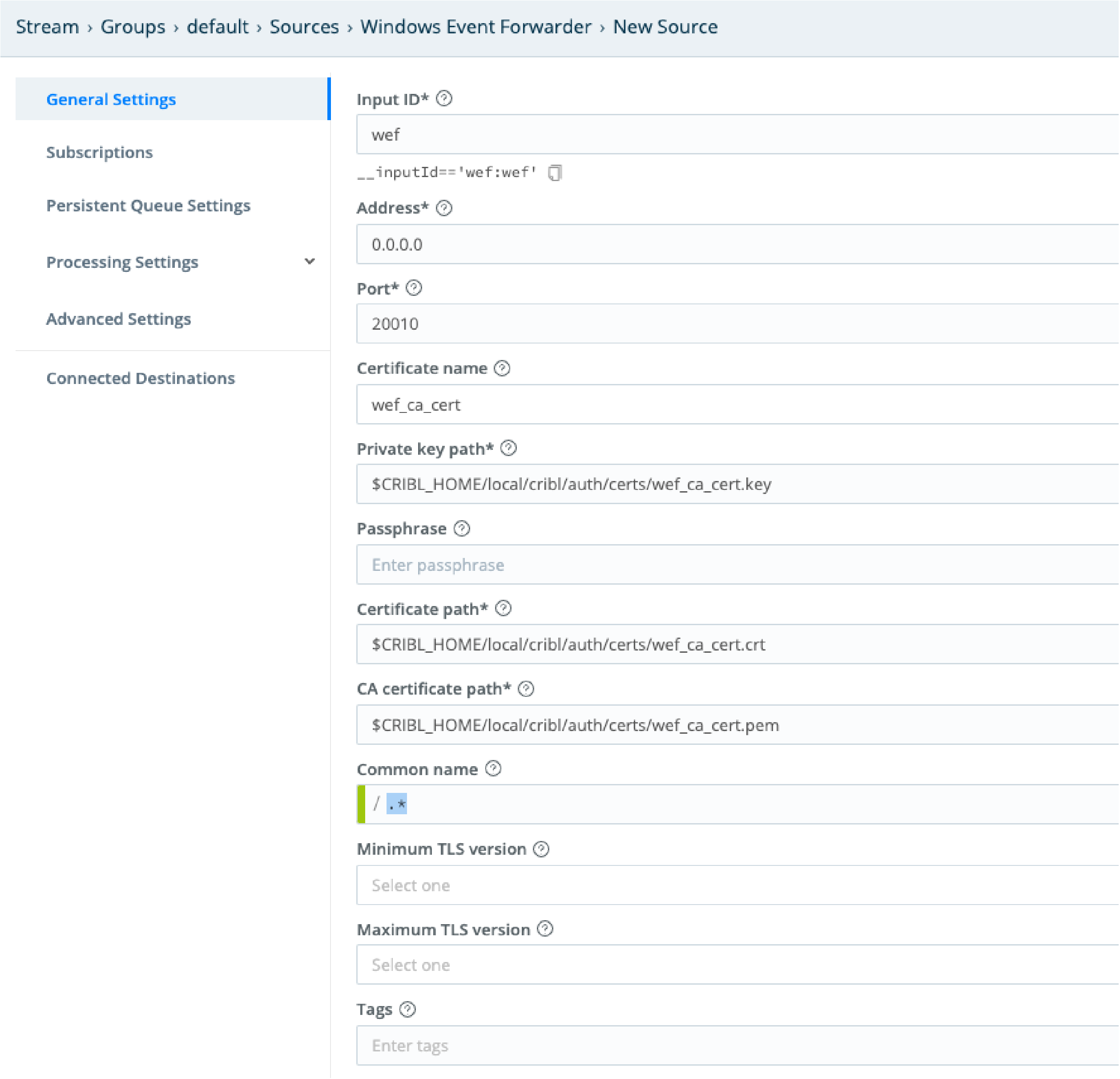
Or, in the **Data Routes** UI: From the top nav of a instance or Group, select **Data** > **Sources**. From the resulting page's tiles or the **Sources** left nav, select [**Push** >] **Windows Event Forwarder**. Next, click **+ Add New** to open a **New Source** modal.

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, enter a custom port between `20000` and `20010`. This is necessary because the default port (`5985`) is not available in Cribl.Cloud.

From the **Certificate name** drop-down, choose the certificate you just created.



Stream › Groups › default › Sources › Windows Event Forwarder › New Source
General Settings
Subscriptions
Persistent Queue Settings
Processing Settings
Advanced Settings
Connected Destinations

Input ID*: wef

`__inputId=='wef:wef'`

Address*: 0.0.0.0

Port*: 20010

Certificate name: wef_ca_cert

Private key path*: $CRIBL_HOME/local/cribl/auth/certs/wef_ca_cert.key

Passphrase: Enter passphrase

Certificate path*: $CRIBL_HOME/local/cribl/auth/certs/wef_ca_cert.crt

CA certificate path*: $CRIBL_HOME/local/cribl/auth/certs/wef_ca_cert.pem

Common name: / .*

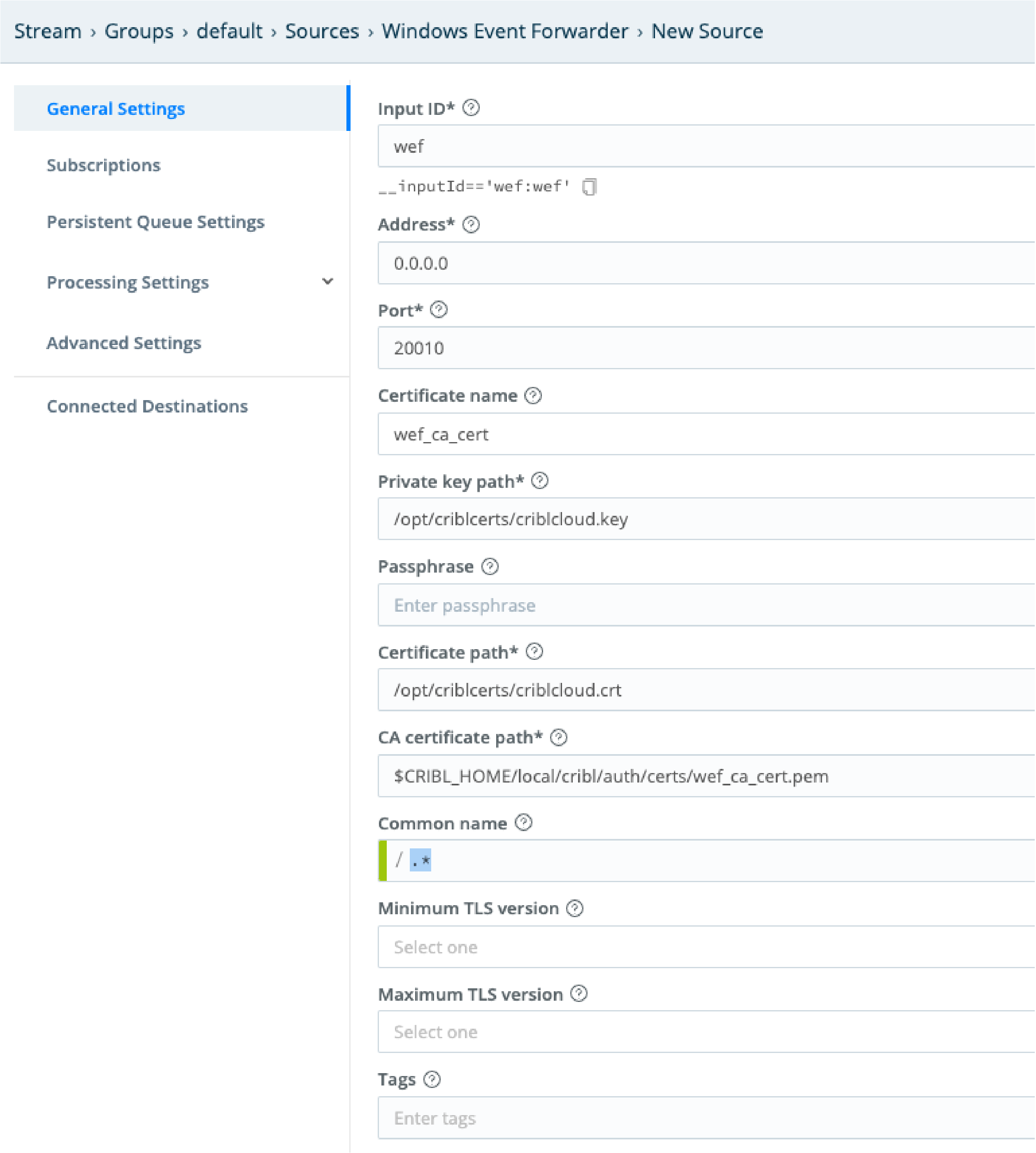
Minimum TLS version: Select one

Maximum TLS version: Select one

Tags: Enter tags

Configuring General Settings

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.)



Stream › Groups › default › Sources › Windows Event Forwarder › New Source
General Settings
Subscriptions
Persistent Queue Settings
Processing Settings
Advanced Settings
Connected Destinations

Input ID*: wef

`__inputId=='wef:wef'`

Address*: 0.0.0.0

Port*: 20010

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: / .*

Minimum TLS version: Select one

Maximum TLS version: Select one

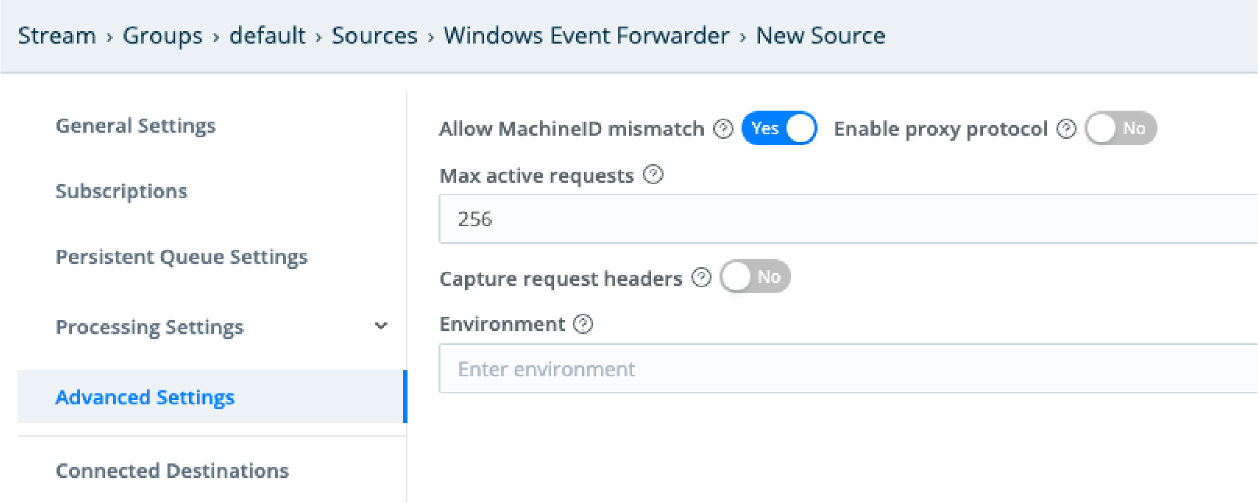
Tags: Enter tags

Configuring General Settings, continued

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 `MachineID`s will match the `Subject Name` specified the certificate. The `No` setting provides better security than `Yes`.


Stream › Groups › default › Sources › Windows Event Forwarder › New Source
General Settings
Subscriptions
Persistent Queue Settings
Processing Settings
Advanced Settings
Connected Destinations

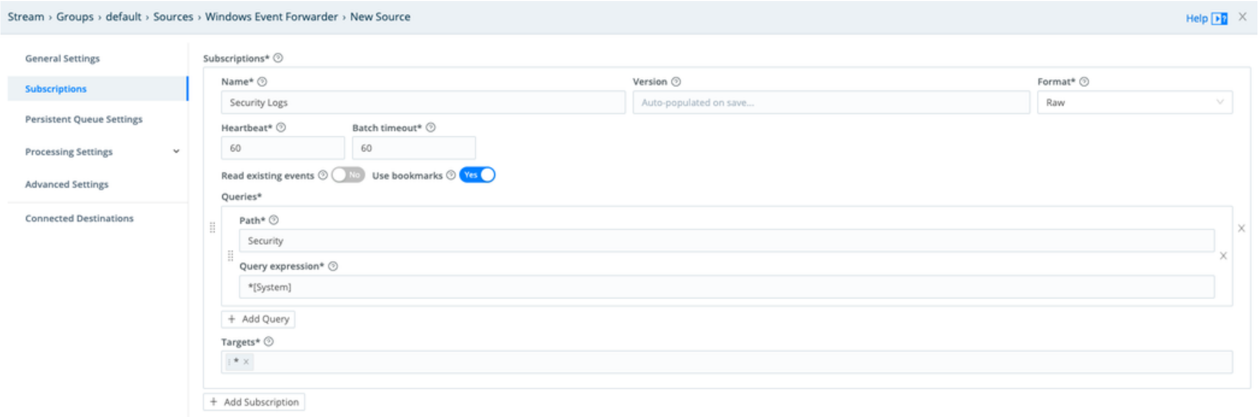
Allow MachineID mismatch: Yes
Enable proxy protocol: No
Max active requests: 256
Capture request headers: No
Environment: Enter environment

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]`.



Subscriptions*

Name*: Security Logs

Version: Auto-populated on save...

Format*: Raw

Heartbeat*: 60

Batch timeout*: 60

Read existing events: No

Use bookmarks: Yes

Queries*

Path*: Security

Query expression*: *[System]

Targets*: *

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.

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.

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.

Cannot find the Certificate

Error:

```
The forwarder is having a problem communicating with subscription manager at address…
The WS-Management service cannot find the certificate that was requested.
```

For example:

```
The forwarder is having a problem communicating with subscription manager at address
https://<stream_worker>:5986/wsman/SubscriptionManager/WEC.

Error code is 2150858882 and Error Message is <f:WSManFault
xmlns:f=http://schemas.microsoft.com/wbem/wsman/1/wsmanfault Code="2150858882"
Machine="DELMD2273309.na.blkint.com"><f:Message>The WS-Management service cannot
find the certificate that was requested. </f:Message></f:WSManFault>.
```

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.

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 querie, see:

- Microsoft's Use Windows Event Forwarding to Help with Intrusion Detection topic.
- The NSA Cybersecurity Directorate's WEF Guidance.
- Palantir's WEF Guidance.

;

12.7. 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.

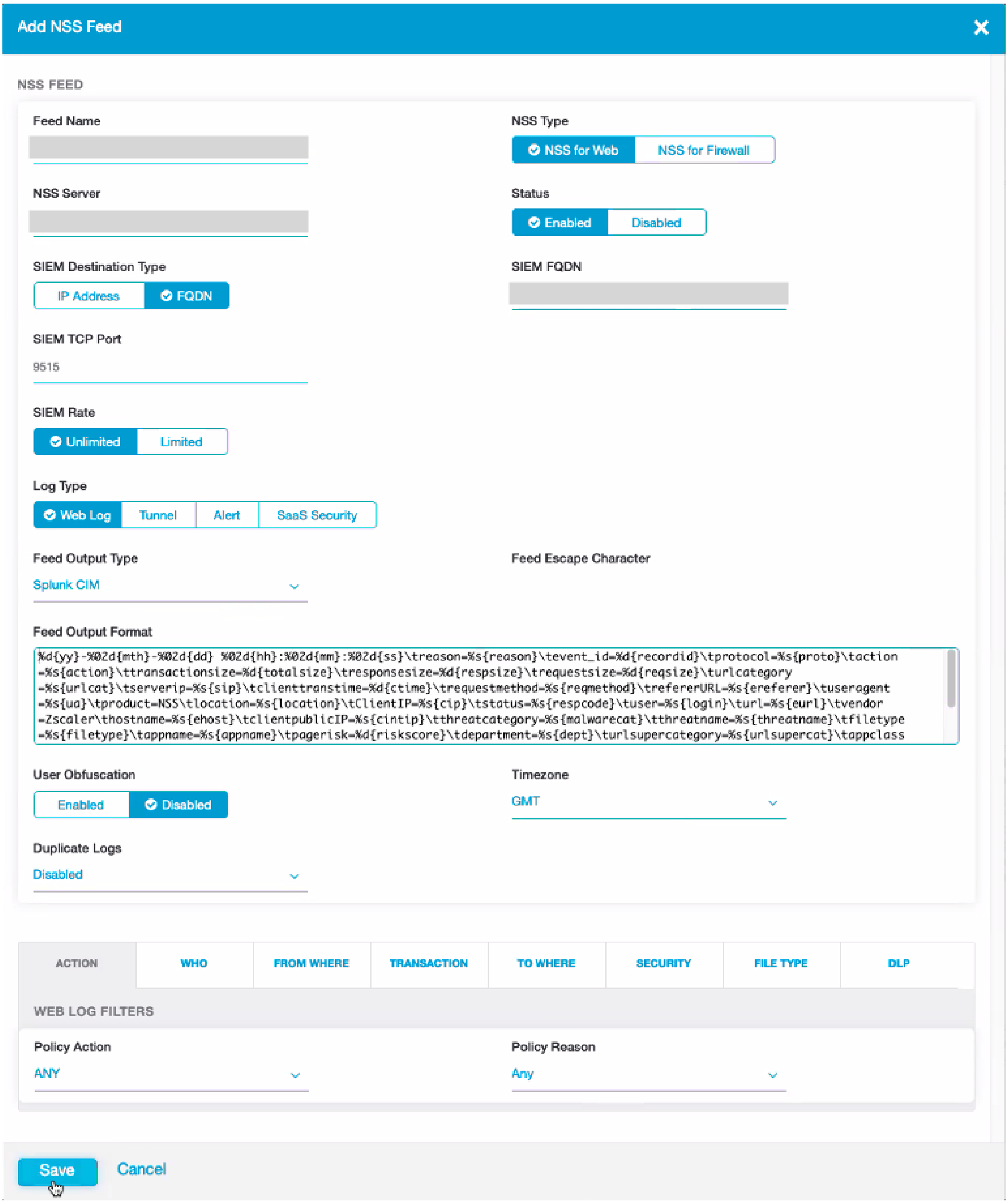
Configuring NSS to Send Data to Cribl Stream

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

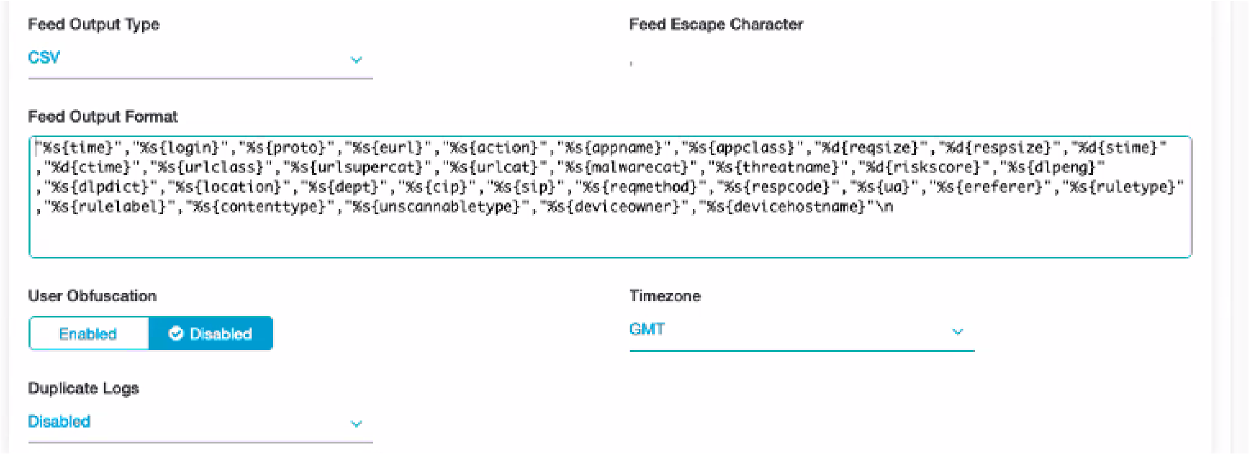
Policy Action
ANY

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

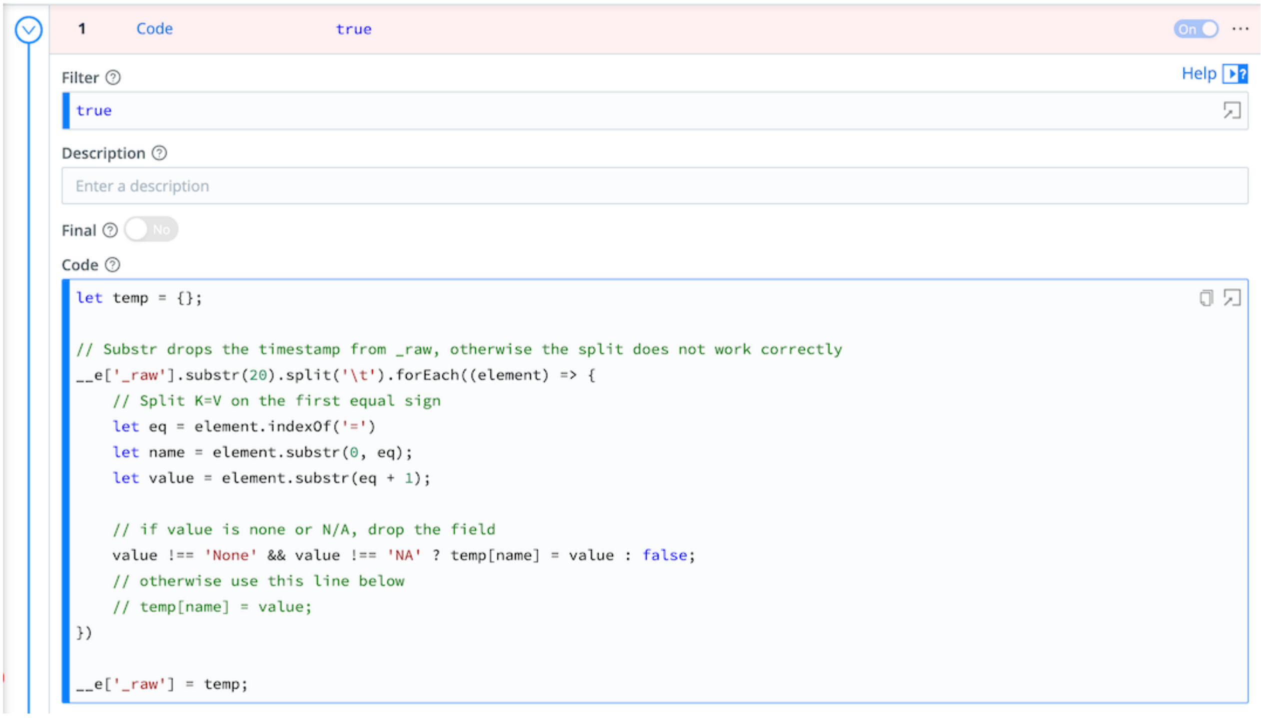
Cribl Stream can reduce Zscaler log size by (1) reformatting and reshaping the data, and (2) suppressing, sampling, and dropping appropriate fields.

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 correctly
__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:



1 Code true

Filter: true

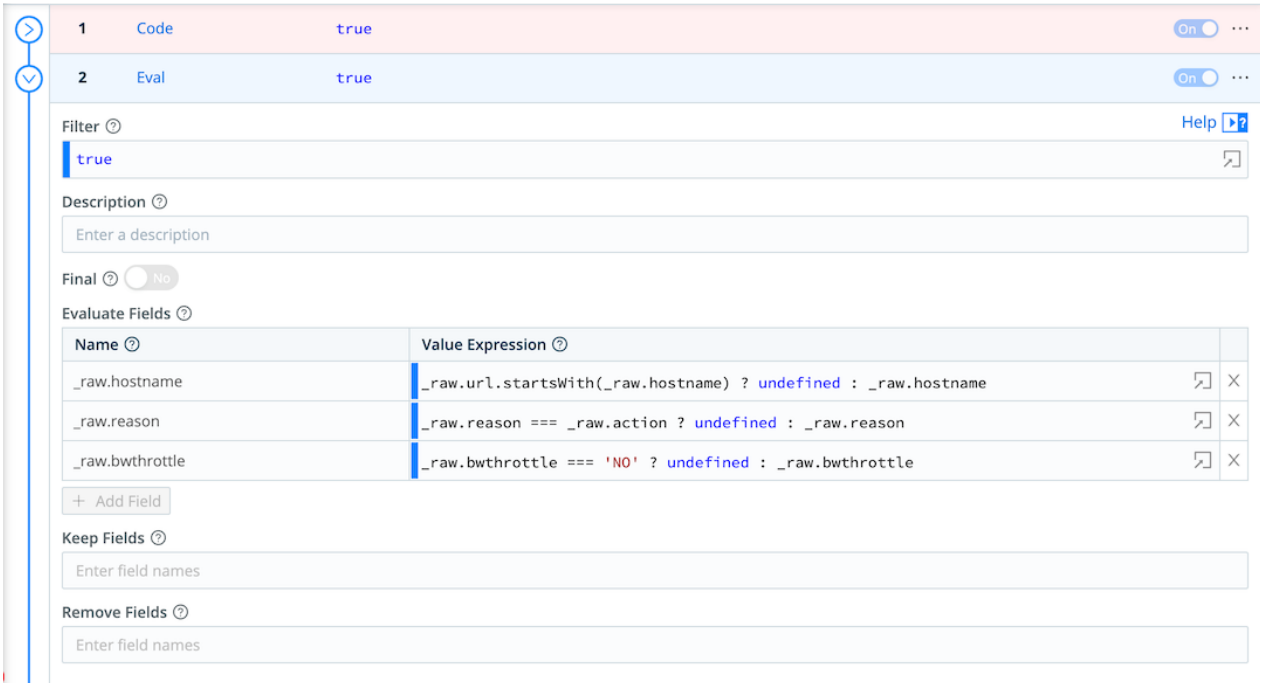
Final: No

Code:

```
let temp = {};

// Substr drops the timestamp from _raw, otherwise the split does not work correctly
__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;
```

Parsing with a Code Function

An Eval Function reshapes the data:



1 Code true

2 Eval true

Filter: true

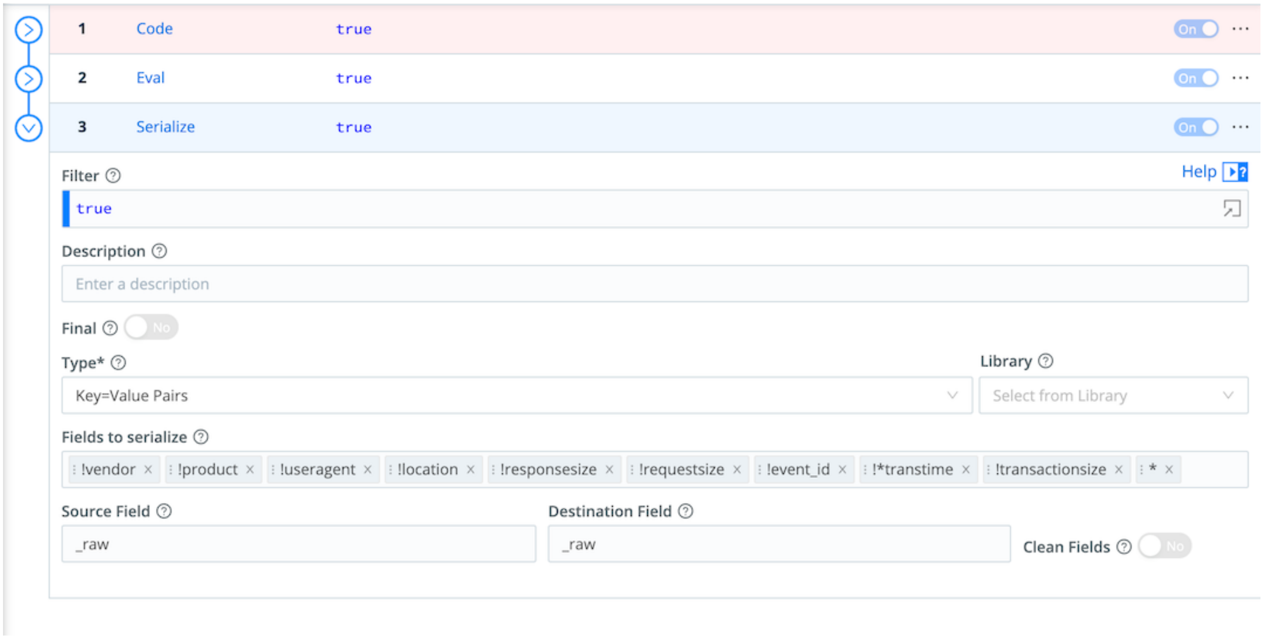
Final: No

Evaluate Fields

Name	Value Expression
_raw.hostname	`_raw.url.startsWith(_raw.hostname) ? undefined : _raw.hostname`
_raw.reason	`_raw.reason === _raw.action ? undefined : _raw.reason`
_raw.bwthrottle	`_raw.bwthrottle === 'NO' ? undefined : _raw.bwthrottle`

Reshaping with Eval

And finally, a Serialize Function drops unwanted fields:



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

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,
otherwise the split does not work
correctly\n__e['_raw'].substr(20).split('\\t').forEach((element) => {\n    // Split
K=V on the first equal sign\n    let eq = element.indexOf('=')\n    let name =
element.substr(0, eq);\n    let value = element.substr(eq + 1);\n\n    // if value
is none or N/A, drop the field\n    value !== 'None' && value !== 'NA' ? temp[name]
= value : false;\n    // otherwise use this line below\n    // temp[name] =
value;\n})\n\n__e['_raw'] = temp;"
        }
      },
      {
        "id": "eval",
        "filter": "true",
        "disabled": false,
        "conf": {
          "add": [
            {
              "name": "_raw.hostname",
              "value": "_raw.url.startsWith(_raw.hostname) ? undefined :
_raw.hostname"
            },
            {
              "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"
        }
      }
    ]
  }
}
```

;

13. Troubleshooting

13.1. Known Issues

This page lists known issues affecting Cribl Stream and/or Cribl Edge.

2022-10-11 – All versions through 3.5.4 – 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: Planned for Cribl Stream 4.0.

2022-08-29 – All versions through 3.5.3 – 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: Planned for Cribl Stream 4.0.1.

2022-08-27 – All versions through 3.5.4 – Azure Event Hubs Destination with PQ drops events when inbound data is interrupted [CRIBL-12649]

Problem: When Azure Event Hubs (and other Kafka-based Destinations) have Persistent Queues 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.

Fix: Version TBD.

2022-08-25 – Version 3.4.1 – 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 the Splunk LB Destination's **Advanced Settings** tab, disable **Minimize in-flight data loss**.

Fix: Version TBD.

2022-08-16 – All versions through 3.5.3 – Splunk HEC shows higher outbound data volume than other Splunk Destinations

Problem: Events sent to the Splunk HEC Destination will show higher outbound data volume than the same events sent to the Splunk Single Instance or Splunk Load Balanced Destinations, which use the S2S binary protocol.

Fix: Planned for Cribl Stream 4.0.1.

2022-08-10 – v.3.5.1 – 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.

2022-08-03 – 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.3.3–3.5.1 – 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) 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 – 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: Version TBD.

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 – v3.5.0–3.5.1 – Cribl Edge/Windows: Upgrading ignores existing mode (etc.) settings [CRIBL-11467]

Problem: When re-running Windows installers on a system that hosts a previous Cribl version as a managed Edge Node, the installer does not read the distributed mode or other settings. The new version might install as a Leader instance.

Workaround: Run the new version's installer using the same mode and other options you used when installing the preceding version.

Fix: Version TBD.

2022-07-19 – v3.5.0–3.5.1 – 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 – v3.5.0–3.5.1 – 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.3.4.2 – 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: Validation check was added in Cribl Stream 3.5.0. Cribl Stream 4.0 and above will skip misnamed metrics when serializing data.

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 3.5.0 – 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 – v3.4.2–3.5.1 – 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 – v3.5.x – Cribl Edge/Windows: Cannot upgrade Edge Nodes from the Leader's UI [CRIBL-10870]

Problem: The specified versions do not support upgrading Edge Nodes via the Leader's UI.

Workaround: Re-run the Windows installer on each Edge Node, specifying the same installation options/parameters you used when installing the preceding version.

Fix: Planned for Cribl Edge 4.0.

2022-06-27 – v.3.3.0–3.5.0 – 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 – 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.

2022-06-01 – v.3.4.2 – 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 – 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: Planned for 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 – 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.

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.3.3.1–3.5.1 – 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: Version TBD.

2022-04-15 – All versions – 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 – 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 – 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.

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: Planned for 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 global ⚙ **Settings** (lower left) > **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 through 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 – versions 3.3.1 through 3.5.0 – 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 – versions 3.2.x through 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.

2022-03-04 – versions 3.3.x through 3.4.0 – Collector jobs fail after upgrade [CRIBL-8516]

Problem: After upgrading a distributed deployment to Cribl Stream 3.3.x or 3.4.0, Collector jobs that use features introduced in the new version might throw errors. This has been observed only on distributed deployments. The root cause is that the Workers are attempting to load the new conig without first reloading the updated schema.

Workaround: Restart the affected Workers. Collector jobs should then work normally.

Fix: Planned for Cribl Stream 3.4.1.

2022-03-03 – versions 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 – versions 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 – versions 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 error`s of the form: `"unable to verify the first certificate"` or `self signed certificate in certificate chain`.

Workaround: On each affected Worker: In the `instance.yml` file's' `distributed: master: tls:` section, set: `"rejectUnauthorized: false"`. Then restart Cribl Stream.

Fix: In 3.4.1.

2022-02-18 – versions 3.3.x – Can't upgrade Workers to 3.3.x via UI [CRIBL-8346]

Problem: Attempting to upgrade on-prem Workers to version 3.3.x via the UI can fail, with an error of the form: `Group <group-name> is a managed group.`

Workaround: Edit the Leader's `$CRIBL_HOME/local/cribl/groups.yml` file to delete all instances of the key-value pair `onPrem: false`. Then, resave the file and reload the Leader.

Fix: In Cribl Stream 3.4.

2022-02-17 – versions 3.2.2 through 3.3.x – REST Collector pagination fails on duplicate/nested attribute names [CRIBL-8352]

Problem: The REST Collector does not consistently differentiate between multiple (nested) attributes that share the same name. This can break pagination that relies on a `Response Body Attribute`.

Workaround: If possible, select a different Pagination method, or specify a unique attribute name.

Fix: In Cribl Stream 3.4.

2022-02-17 – v.3.3.0 – Do not configure InfluxDB Destination on LogStream 3.3.0 [CRIBL-8354]

Problem: Configuring InfluxDB Destinations can fail on LogStream 3.3.0. This problem has been observed only on distributed deployments, and data does flow to InfluxDB. But the InfluxDB config will not appear in LogStream's UI, and the broken config will block editing of other Destinations.

Workaround: If you need to send data to InfluxDB, skip v.3.3.0, and wait for the next release. If you encounter the broken UI, edit `outputs.yml` to remove any `influxdb_output` sections, and then restart LogStream. This will unblock UI-based editing of other Destinations.

Fix: In LogStream 3.3.1.

2022-02-10 – All versions, 3.2.1+ – Syslog Source's previewed data doesn't proceed through Routes [CRIBL-8210]

Problem: The Syslog Source's **General Settings** > **Input ID** field's default filter expression is `__inputId.startsWith('syslog:in_syslog:')`. The same filter appears in the **Preview Full** tab's **Entry Point** drop-down, except without the final colon. This means that data sent using **Preview Full**'s version of the filter will not go through Routes that use the **Input ID** version.

Workaround: When sending data from **Preview Full** to a Route, if both use the `syslog:in_syslog` filter, edit the Route's filter to remove the final colon. This will make the filter identical in both places, routing data correctly.

2022-02-10 – version 3.3 – 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.

2022-02-08 – versions 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.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 – 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 – versions 3.2.2 through 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 – 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 – All 3.x versions – 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: Planned for Cribl Stream 4.0.

2022-01-11 – v.3.2.2 – Git Collapse Actions disaggregates [CRIBL-7638]

Problem: With **Git Settings** > **Collapsed actions** enabled, the combined **Commit & Push** button appears in the modal as expected, but then saving a new change splits it back into two separate buttons: **Commit** and **Deploy**.

Workaround: Disable the **Collapse actions** setting, then commit and deploy separately.

Fix: In LogStream 3.3.

2022-01-19 – v.3.2.2 – Usernames containing certain letters cause some API requests to fail [CRIBL-7728]

Problem: Usernames containing certain letters can cause some API requests to fail with an `Invalid character in header content ["cribl-user"]` error. This can happen even when the username is valid for an Identity Provider.

Workaround: In your usernames, make sure that the letters you use are limited to the letters in Latin alphabet no. 1 as defined in the ISO/IEC 8859-1 standard.

Fix: In LogStream 3.3.

2022-01-18 – versions 3.2.0 through 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: Planned for Cribl Stream 3.4.1.

2022-01-04 – v.3.2.x – Enabling GitOps deletes LogStream's `bin`, `logs`, and `pid` subdirectories [CRIBL-7513]

Problem: Enabling GitOps with LogStream 3.2.x, via either CLI or UI, can cause the deletion of the `bin`, `logs`, and `pid` subdirectories. This disables LogStream. The root cause is that `git` versions 2.13.1 and lower did not fully respect paths listed in `.gitignore`.

Workaround: Upgrade your `git` client to v.2.13.2 or higher, to correct the underlying behavior.

Fix: In LogStream 3.3.

2022-01-03 – v.3.2.1 Leader cannot upgrade pre-v.3.2.0 Workers [CRIBL-7498]

Problem: A 3.2.1 Leader's UI cannot upgrade Workers running LogStream 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`.

Workaround: Upgrade the Workers via the filesystem.

Fix: Does not affect Workers running Cribl Stream (LogStream) 3.2.0 or higher.

2021-12-21 – v.3.2.2 through 3.5.0 – 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 3.5.2.

2021-12-15 – v.3.2.1 through 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 through 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.

2021-12-09 – All versions through 3.2.1 – Worker's Config Version value spins indefinitely [CRIBL-7306]

Problem: For a newly created Worker Group, the Workers page's **Config Version** column can show an indefinitely spinning progress spinner for that Group's Workers. This happens because Workers always expect a config bundle to be available to download from the Leader, but no config has been deployed yet.

Workaround: Click the **Deploy** option to deploy a config to the affected Workers' Group.

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 – v3.1.2, 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 – v3.2.0 – 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**.

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 – v3.2.0 – 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 – v3.2.0 – 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 – v3.2.0 – Exporting a Pack to another Group requires a Leader restart [CRIBL-7043]

Problem: Exporting a Pack to a different Worker Group via the UI succeeds, but opening the Pack on the target Group fails with a `Cannot read property...undefined` error.

Workaround: To resolve the error and make the target Pack accessible, restart the Leader. To prevent the error, export and import the Pack as a file.

Fix: In LogStream 3.2.1.

2021-11-17 – versions 2.1 through 3.2.1 – 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: Planned for Cribl Stream 3.4.

2021-11-16 – v3.2.0 – 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.

2021-10-11 – versions 2.x through 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 – v3.1.2 – Monitoring page omits Collector Sources' data [CRIBL-6412]

Problem: With functioning Office 365 Activity and Office 365 Services Sources, the Job Inspector reports data being retrieved, and **Monitoring** > **Data** > **Destinations** reports data being sent out. However, **Monitoring** > **Data** > **Sources** falsely reports no data being received. The problem is isolated to this **Sources** page. It might also affect the Office 365 Message Trace and Prometheus Scraper Sources.

Workaround: Use the Source modal's **Live Data** tab, the **Monitoring** > **System** > **Job Inspector** page, and/or the **Monitoring** > **Data** > **Destinations** page to monitor throughput.

Fix: In LogStream 3.1.3.

2021-10-02 – All v.3.2.x – 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: Planned for Cribl Stream 3.4.

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, 3.1.1 – 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 – v3.1.1 – 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.

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 – 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 v.3.0.4.

Fix: Upgrade to v.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: Planned for LogStream 3.1.1, but couldn't reproduce the error.

2021-07-29 – v.3.0.2 through 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: Planned for LogStream 3.1.1, but couldn't reproduce the error.

2021-07-20 – v.3.0.3 – 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 3.1.x – 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: Planned for Cribl Stream 3.4.

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 – 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.

2021-06-22 – v.3.0.0–3.0.2 – 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 3.0.3.

2021-05-20 – 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: In LogStream 3.0.1.

2021-05-19 – 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 – 3.0.0 – Packs > Export in "Merge" mode omits schemas and custom Functions

Problem: Exporting a Pack with the export mode set to `Merge` omits schemas and custom Functions configured within the Pack's **Knowledge > Schemas**.

Workarounds: 1. Change the export mode to `Merge safe`, and export again. 2. If that doesn't preserve the schema and Functions, revert to `Merge` export mode; install the resulting Pack onto its target(s); and then manually copy/paste the schema(s) and Functions from the source Pack's UI to the target Pack's UI.

Fix: In LogStream 3.0.1.

2021-05-17 – v.3.0.0–3.2.0 – Can't Enable KMS on Worker Group after installing license

Problem: Enabling HashiCorp Vault or AWS KMS on a Worker Group, after installing a LogStream license on the same Group, fails with a spurious `External KMS is prohibited by the current license` error message.

Workaround: On the Leader, navigate to **Settings** > **Worker Processes**. Restart the affected Worker Group's `CONFIG_HELPER` process. Then return to that Worker Group's **Security** > **KMS** Settings, re-enter the same KMS configuration, and save.

2021-05-10 – 2.4.5 – 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 – 2.4.5 – 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.2.4.4–3.01 – 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 – 2.4.4 – 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 – 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 – 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 through 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 – 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, 2.4.1, 2.4.2 – Splunk (S2S) Forwarder access control blocks upon upgrade to LogStream 2.4.x

Problem: If Splunk indexers have forwarder tokens enabled, and worked with LogStream 2.3.x before, upgrading to LogStream 2.4.x causes data to stop flowing.

Workaround: If you encounter this problem, rolling back to your previously installed LogStream version (such as v.2.3.4) resolves it.

Fix: In LogStream 2.4.3.

2021-02-10 – v.2.4.0, 2.4.1 – With Splunk HEC Source, JSON payloads containing embedded objects trigger high CPU usage

Problem: Splunk HEC JSON payloads containing nested objects trigger high CPU usage, due to a flaw in JSON parsing.

Workaround: If you encounter this problem, rolling back to your previously installed LogStream version (such as v.2.3.4) resolves it.

Fix: In LogStream 2.4.2.

2021-01-30 – v.2.4.0 – Worker Nodes cannot connect to Master

Problem: Worker Nodes cannot connect to the Master after the Master is upgraded to v.2.4.0.

Workaround: Disable compression on the Workers. You can do so through the Workers' UI at **System Settings > Distributed Settings > Master Settings > Compression**, or by commenting out this line in each Worker's `cribl.yml` config file:

```
compression: gzip
```

Fix: In LogStream 2.4.1.

2021-01-25 – v.2.4.0 – S3 collection stops working due to auth secret key issues.

Problem: S3 collection stops after upgrade to 2.4.0 due to secret key re-encryption.

Workaround: Re-configure S3, save and re-deploy.

Fix: In LogStream 2.4.1.

2021-01-14 – v.2.4.0 – Google Cloud Storage Destination Needs Extra Endpoint to Initialize

Problem: The Google Cloud Storage Destination fails to initialize, displaying an error of the form: `Bucket does not exist!`

Workaround: In the `outputs.yml` file, under your `cribl-gcp-bucket` key endpoint, add: `https://storage.googleapis.com`. (in a single-instance deployment, locate this file at `$CRIBL_HOME/local/cribl/outputs.yml`. In a distributed deployment, locate it at `$CRIBL_HOME/groups/<group name>/local/cribl/outputs.yml`.)

Fix: In LogStream 2.4.1.

2021-01-14 – v.2.4.0 – Worker Groups' Settings > Access Management Is Absent from UI

Problem: In this release, the **Worker Groups > <group-name> > System Settings** UI did not display the expected **Access Management**, **Authentication**, and **Local Users** sections.

Workaround: Manually edit the `users.json` file.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – Route Filters Aren't Copied to Capture Modal

Problem: On the **Routes** page, selecting **Capture New** in the right pane does not copy custom **Filter** expressions to the resulting **Capture Sample Data** modal. That modal's **Filter Expression** field always defaults to `true`.

Workarounds: 1. Bypass the **Capture New** button. Instead, from the Route's own ••• (Options) menu, select **Capture**. This initiates a capture with the **Filter Expression** correctly populated. 2. Copy/paste the expression into the **Capture Sample Data** modal's **Filter Expression** field. Or, if the expression is displayed in that field's history drop-down, retrieve it.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – Destinations' Documentation Doesn't Render from UI

Problem: Clicking the **Help** Help ▶? link in a Destination's configuration modal displays the error message: "Unable to load docs. Please check LogStream's online documentation instead."

Workarounds: 1. Go directly to the online Destinations docs, starting here. 2. Follow the UI link to the docs landing page, click through to open or download the current PDF, and scroll to its Destinations section.

Fix: In LogStream 2.4.1.

2021-01-13 – v.2.4.0 – `Esc` Key Doesn't Consistently Close Modals

Problem: Pressing `Esc` with focus on a modal's drop-down or slider doesn't close the modal as expected. (Pressing `Esc` with focus on a free-text field, combo box, or nothing does close the modal – displaying a confirmation dialog first, if you have unsaved changes.)

Workarounds: Click the **X** close box at upper right, or click **Cancel** at lower right.

Fix: In LogStream 2.4.1.

2020-12-17 – v.2.3.0+ – 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 – In-product upgrade issue on v2.0

Problem: Using in-product upgrade feature in v.1.7 (or earlier) fails to upgrade to v2.0, 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 v1.7

Problem: Using in-product upgrade feature in v1.6 (or earlier) fails to upgrade to v1.7 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 v1.2 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.

;

13.2. 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
- Intercom chat button
- Community Slack
- Support Portal (login required)

The best method depends on your reason for contacting Support. For quick, question-and-answer discussions, Community Slack or Intercom chat might be sufficient. For in-depth discussions, or for troubleshooting 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.

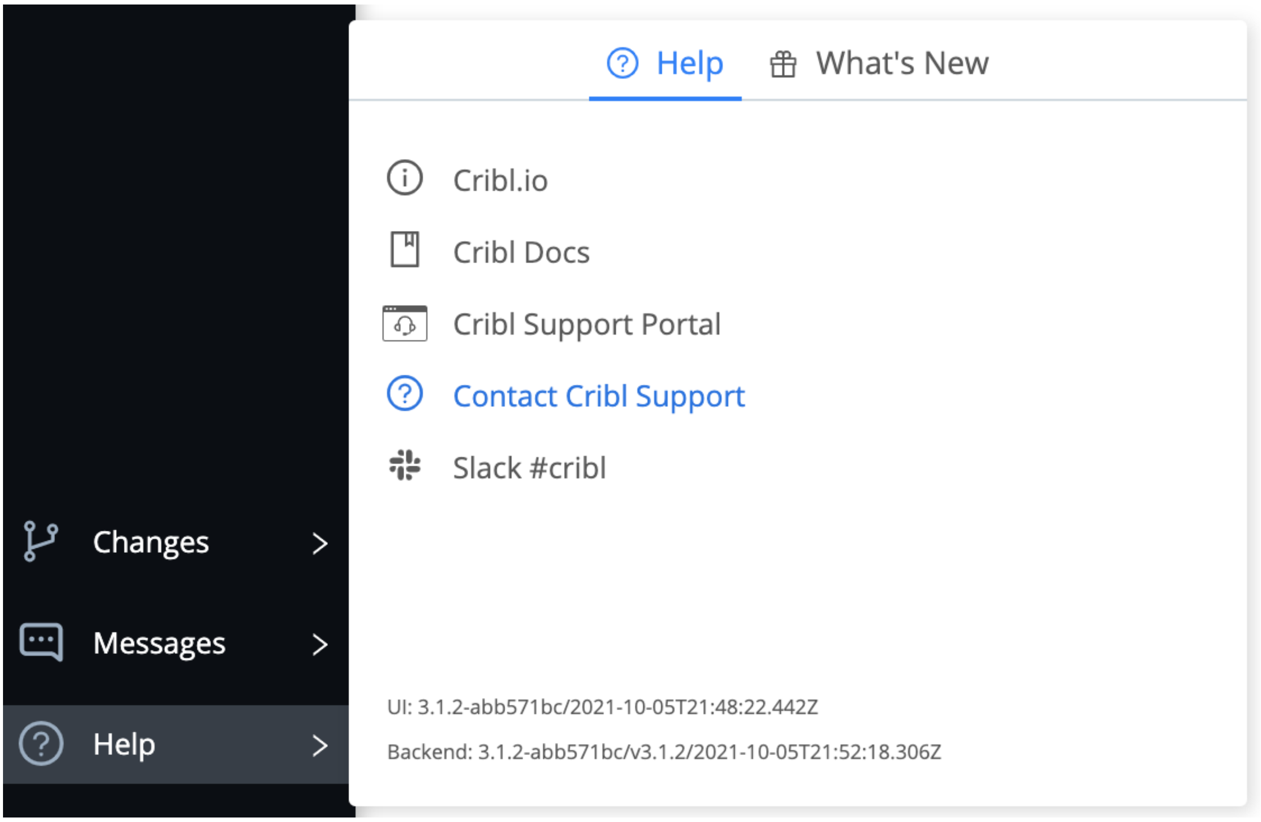
Email communication imposes a total message-size limit of 25 MB (after MIME-encoding attachments). The Support Portal allows file attachments of up to 100 MB each.

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 Help Button

Within Cribl Stream's UI, you can click the left nav's **(?) Help** link and, from the resulting fly-out, select **Contact Support**. This will prompt creation of a new email in your email client.



UI: 3.1.2-abb571bc/2021-10-05T21:48:22.442Z

Backend: 3.1.2-abb571bc/v3.1.2/2021-10-05T21:52:18.306Z

Email from the UI

Contact via Intercom Button

Within Cribl Stream's UI, you can click the Intercom button at the bottom right to send Cribl Support questions and (if necessary) screenshots and other files.

We recommend Intercom only for quick questions/answers, because it doesn't provide robust tracking of communications or files.

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 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 Launch Guide.

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.

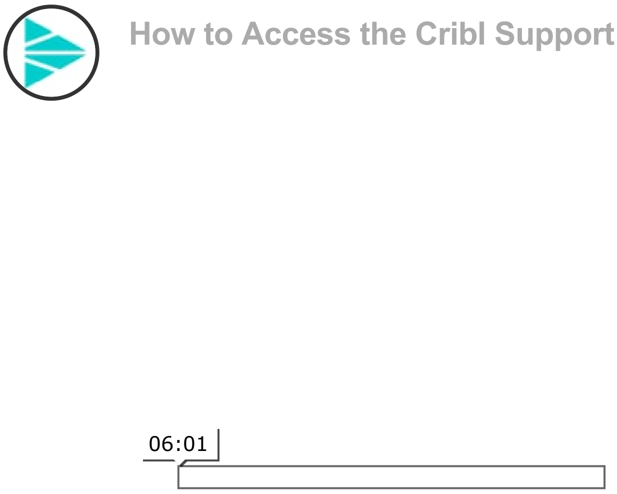
- Invite Standard users to the portal.

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 (i.e., no shared or group accounts). 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 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, the diag is uploaded from the browser, rather than from the LogStream Node. This means that your LogStream 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 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 through Intercom or 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.

Peer Support: Cribl Curious Q&A Site

The Cribl Curious Q&A site is a searchable slate of frequently asked questions, with answers from expert users and Cribl insiders. If you can't find the answer to your question – post it here. If you can help out fellow users – earn goodwill and points.

;

13.3. 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 or distributed), you can create and **securely** share bundles with Cribl Support either from the UI or from the CLI. In either case, you'll need outbound internet access to https://diag-upload.cribl.io and a valid Support Case number. That site 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.)

With a Cloud deployment, contact Cribl Support to gather a diag bundle on your behalf.

Using the UI

To create a bundle, go to global ⚙ **Settings** (lower left) > **Diagnostics > Diagnostic Bundle** and click **Create Diagnostic Bundle**.

- To download the bundle locally to your machine, click **Export**.

- To share the bundle with Cribl Support, toggle **Send to Cribl Support** to **Yes**, enter your case number, and then click **Export**.

You can create a bundle from individual workers if you have the Worker UI access setting enabled. Go to **Workers >** <worker-name> > **Settings** (top right) > **Diagnostics > Diagnostic Bundle**, and click **Create Diagnostic Bundle**.

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.

Using the CLI

To create a bundle using the CLI, use the `diag` command.

```
# $CRIBL_HOME/bin/cribl diag
Usage: [sub-command] [options] [args]

Commands:
get    - List existing Cribl Stream diagnostic bundles
create - Creates diagnostic bundle for Cribl Stream
send   - Send Cribl Stream diagnostic bundle to Cribl Support, args:
   -c <caseNumber> - Cribl Case Number
  [-p <path>]      - Diagnostic bundle path (if empty, then new bundle will be
created)


## Creating a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag create
Created Cribl Stream diagnostic bundle at /opt/cribl/diag/cribl-logstream-
<hostname>-<datetime>.tar.gz.

## Creating and sending a diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -c 420420
Sent Cribl Stream diagnostic bundle to Cribl Support

## Sending a previously created diagnostic bundle
# $CRIBL_HOME/bin/cribl diag send -p /opt/cribl/diag/cribl-logstream-<hostname>-
<datetime>.tar.gz -c 420420
Sent Cribl Stream diagnostic bundle to Cribl Support
```

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 PIDs consuming a lot of CPU.

2. See Sizing & Scaling > CPU Profiling 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 3–6 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

If you encounter a memory leak, Cribl Support might request a memory snapshot of a Worker Process, to better understand where the leak is occurring. To capture the memory snapshot, follow these instructions.

Configure Node JS Debug Mode

On the affected system, stop the LogStream process or service.

Systemd

If you have bootstrapped Cribl Stream, follow the section for Persisting Systemd Overrides. Add the following to the `[Service]` stanza:

```
Environment=NODE_ENV=debug
```

Once the environment variable has been added to the override file, run the following commands to start Cribl Stream again:

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

Command Line

If starting Cribl Stream from the CLI, prepend `NODE_ENV=debug` to your command. E.g.: `NODE_ENV=debug $CRIBL_HOME/bin/cribl start`

Find Inspect Ports

On the Leader or Worker that you have configured for debug mode, run the following command: `ps aux | grep cribl | grep -v grep`. This will generate an output similar to the following:

```
[root@worker ~]# ps aux | grep cribl | grep -v grep
cribl    11525  2.6  1.4 1021944 150988 ?      Sl   07:36   0:25
/opt/cribl/bin/cribl server
cribl    11539 20.3  1.9 1136336 201164 ?      Rl   07:36   3:21
/opt/cribl/bin/cribl --inspect=0.0.0.0:9230 --inspect-port=9230
/opt/cribl/bin/cribl.js server -r WORKER
cribl    11545 20.3  2.0 1134336 204676 ?      Sl   07:36   3:21
/opt/cribl/bin/cribl --inspect=0.0.0.0:9230 --inspect-port=9231
/opt/cribl/bin/cribl.js server -r WORKER
```

Make note of the `--inspect-port=` values.

In the next step, you will configure Google Chrome to connect to these ports. You will need to ensure network connectivity from your machine to the Cribl Stream instance's inspect ports.

> If you don't have direct network connectivity from your machine to the Cribl Stream instance, you will need to use SSH port forwarding. Repeat this example command for each port that you need to forward:
>
> ```
> ssh -L 9230:localhost:9230 cribl@example.com
> ```

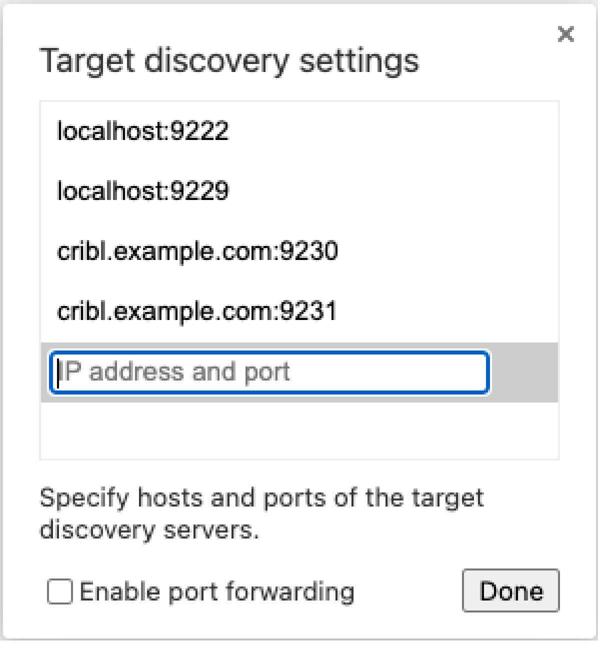
Obtain Remote Snapshots Using Google Chrome

Open Google Chrome and navigate to chrome://inspect. Ensure that you are on the Devices tab.

To the right of **Discover network targets**, click the **Configure...** button.

If you have previously generated a memory snapshot using this method, delete the old hosts from this list.

Next, as shown in the example below, enter the host and port combinations that you obtained in the previous step.



Target discovery settings

localhost:9222

localhost:9229

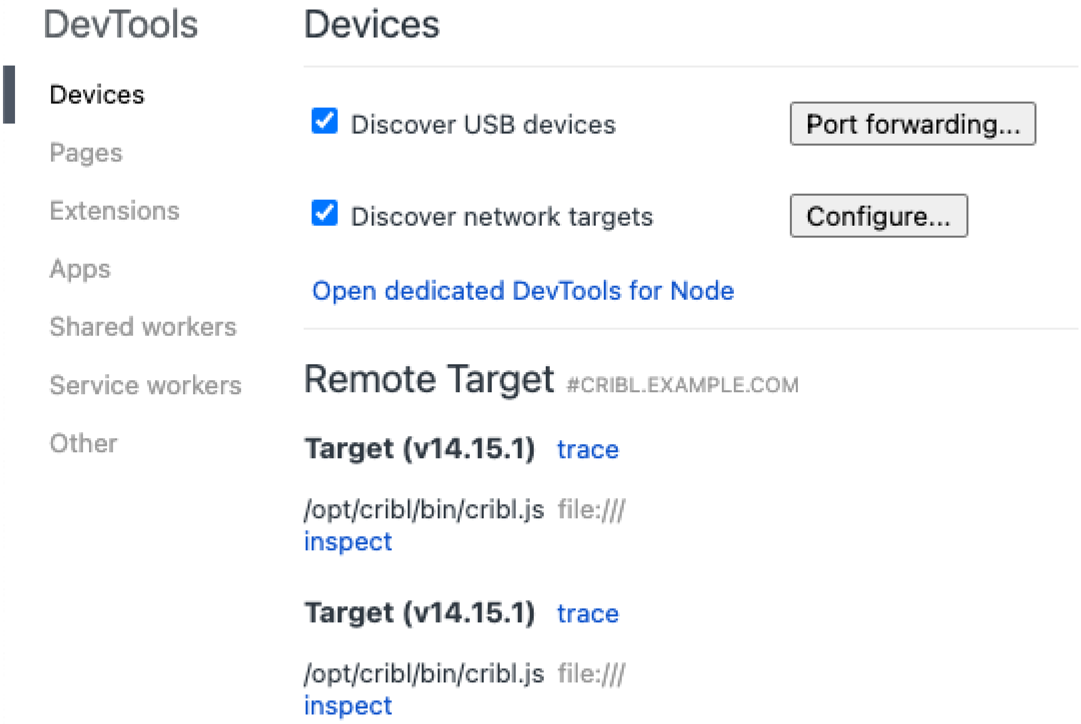
cribl.example.com:9230

cribl.example.com:9231

Specify hosts and ports of the target discovery servers.

☐ Enable port forwarding

Click the **Done** button. You will now have a list of **Remote Target**s that you can inspect.



Devices

☑ Discover USB devices

☑ Discover network targets

Remote Target #CRIBL.EXAMPLE.COM

Target (v14.15.1) trace

/opt/cribl/bin/cribl.js file:///
inspect

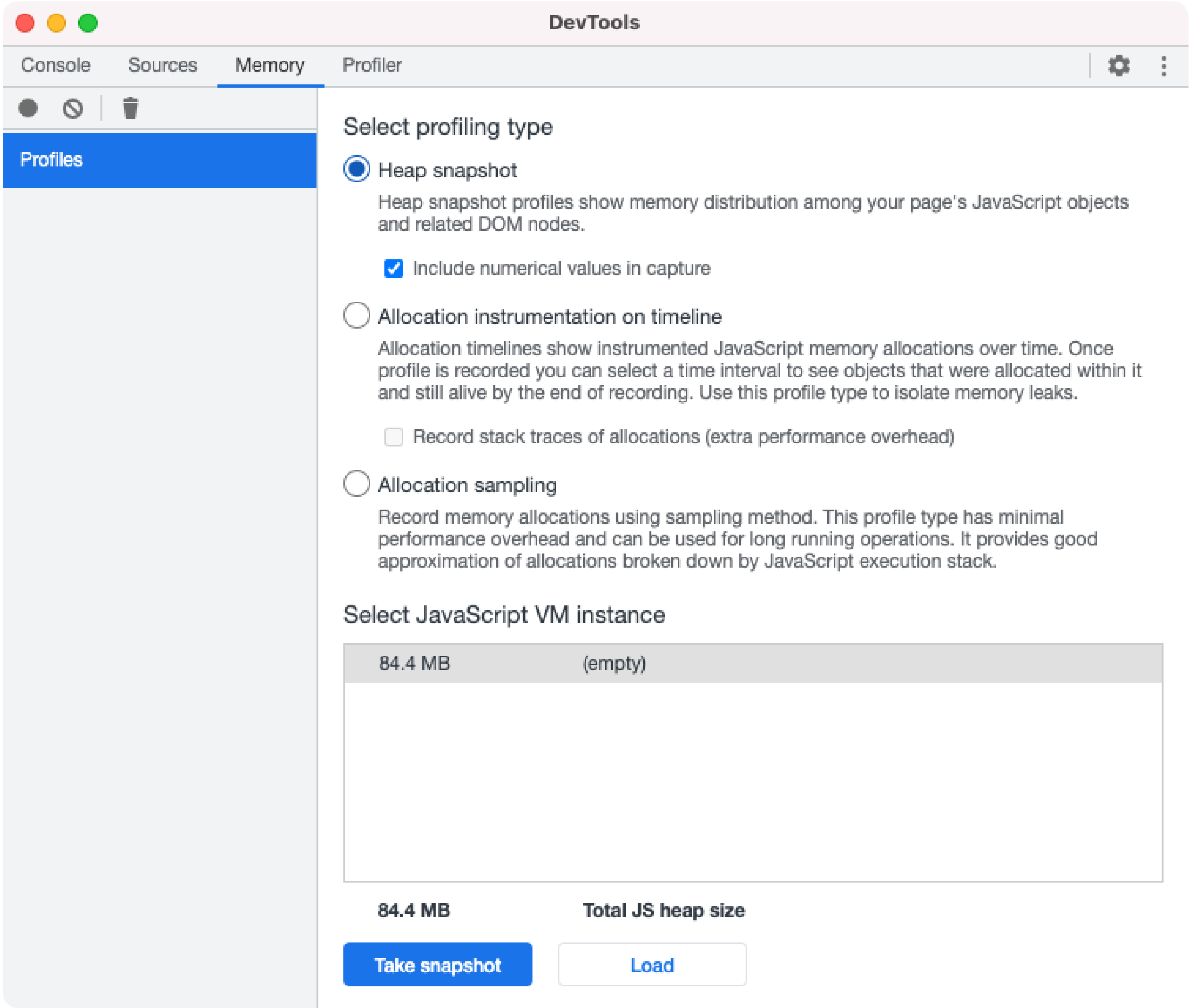
Target (v14.15.1) trace

/opt/cribl/bin/cribl.js file:///
inspect

Click the blue **inspect** link for each target. This will open a new window. Select its **Memory** upper tab.

Select **Heap snapshot**, and select the check box labeled **Include numerical values in capture**.

Next, click the blue **Take snapshot** button at the bottom of the window.



Select profiling type

● Heap snapshot

Heap snapshot profiles show memory distribution among your page's JavaScript objects and related DOM nodes.

☑ Include numerical values in capture

○ Allocation instrumentation on timeline

Allocation timelines show instrumented JavaScript memory allocations over time. Once profile is recorded you can select a time interval to see objects that were allocated within it and still alive by the end of recording. Use this profile type to isolate memory leaks.

□ Record stack traces of allocations (extra performance overhead)

○ Allocation sampling

Record memory allocations using sampling method. This profile type has minimal performance overhead and can be used for long running operations. It provides good approximation of allocations broken down by JavaScript execution stack.

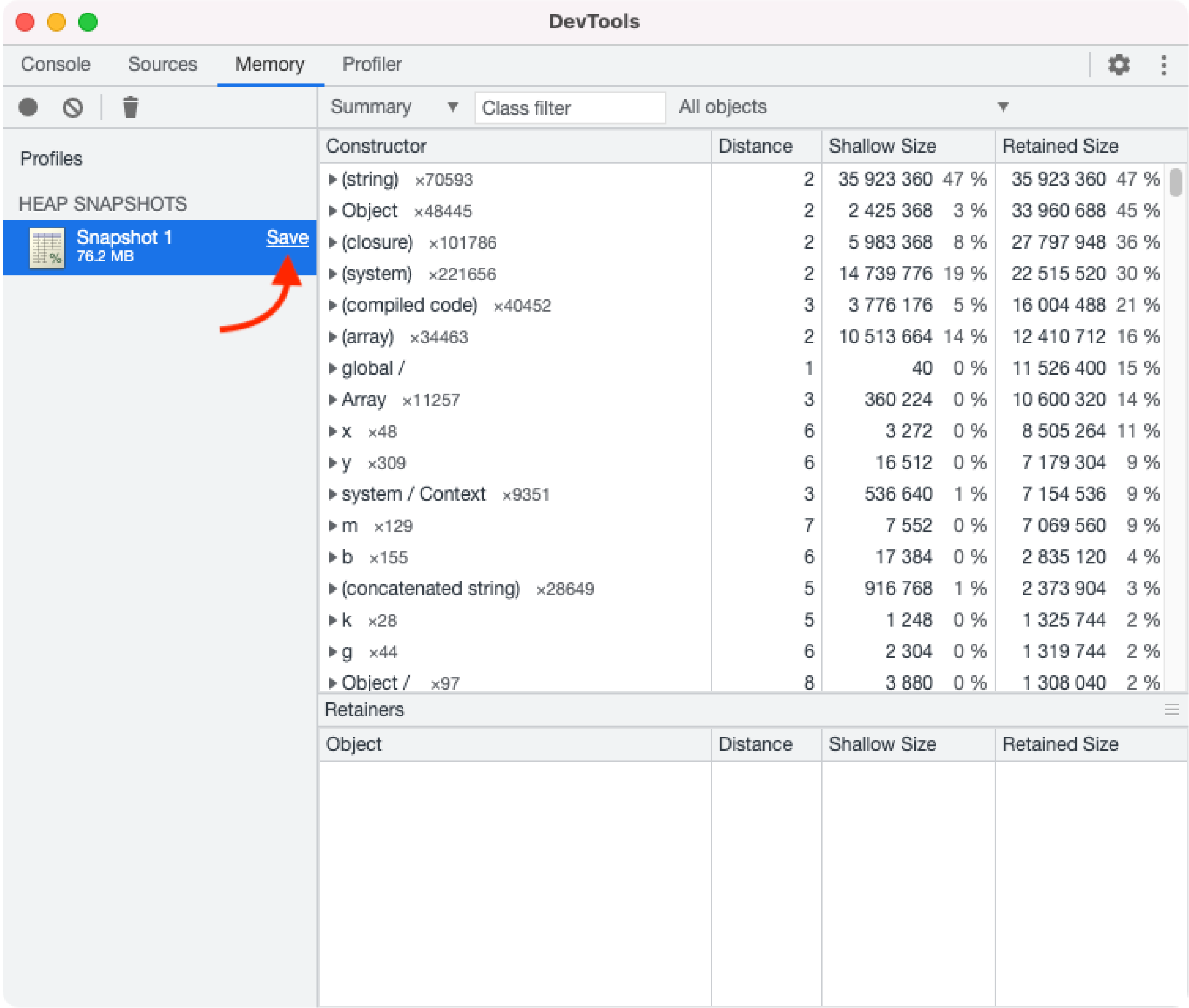
Select JavaScript VM instance

84.4 MB	(empty)

84.4 MB **Total JS heap size**

Download each snapshot by clicking the **save** link to the right of the snapshot number. As you download these files, be sure to append the Worker Process or inspect port number to each file name.

Then provide these files to Cribl Support.



Constructor	Distance	Shallow Size	Retained Size
▸(string) ×70593	2	35 923 360 47 %	35 923 360 47 %
▸Object ×48445	2	2 425 368 3 %	33 960 688 45 %
▸(closure) ×101786	2	5 983 368 8 %	27 797 948 36 %
▸(system) ×221656	2	14 739 776 19 %	22 515 520 30 %
▸(compiled code) ×40452	3	3 776 176 5 %	16 004 488 21 %
▸(array) ×34463	2	10 513 664 14 %	12 410 712 16 %
▸global /	1	40 0 %	11 526 400 15 %
▸Array ×11257	3	360 224 0 %	10 600 320 14 %
▸x ×48	6	3 272 0 %	8 505 264 11 %
▸y ×309	6	16 512 0 %	7 179 304 9 %
▸system / Context ×9351	3	536 640 1 %	7 154 536 9 %
▸m ×129	7	7 552 0 %	7 069 560 9 %
▸b ×155	6	17 384 0 %	2 835 120 4 %
▸(concatenated string) ×28649	5	916 768 1 %	2 373 904 3 %
▸k ×28	5	1 248 0 %	1 325 744 2 %
▸g ×44	6	2 304 0 %	1 319 744 2 %
▸Object / ×97	8	3 880 0 %	1 308 040 2 %

Object	Distance	Shallow Size	Retained Size

Incomplete File Names?

If you forgot to record the Worker Process from which snapshots originated, you can run the following command:

```
grep -r '/opt/cribl/state/w_' .
```

Example output:

```
./Heap-20220222T113432.heapsnapshot:"/opt/cribl/log/worker/2/cribl.log",
./Heap-20220222T115411.heapsnapshot:"/opt/cribl/log/worker/4/cribl.log",
./Heap-20220222T111147.heapsnapshot:"/opt/cribl/log/worker/1/cribl.log",
```

The above files correspond to Worker Processes 2, 4, and 1, respectively.

;

13.4. 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.

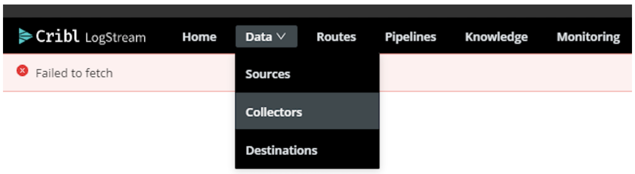
Web UI

Where: In Cribl Stream's UI.

Error: "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.



Failed to fetch

Failed to fetch

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 its **Worker Groups** menu (2.x), or **Groups** or **Configure** (3.x) menu, before the Cribl Stream server is up.

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: On the Leader Node, manually edit `kms.yml` to use `provider: local`. Then restart Cribl Stream to correct the path conflict.

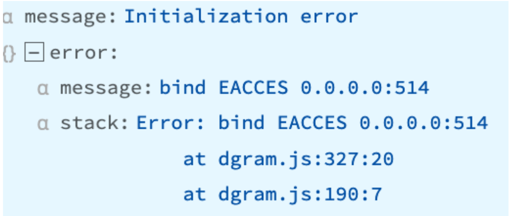
Sources (General)

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.



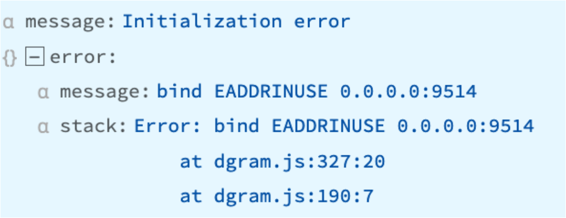
```
message: Initialization error
error:
  message: bind EACCES 0.0.0.0:514
  stack: Error: bind EACCES 0.0.0.0:514
             at dgram.js:327:20
             at dgram.js:190:7
```

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.



```
message: Initialization error
error:
  message: bind EADDRINUSE 0.0.0.0:9514
  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.

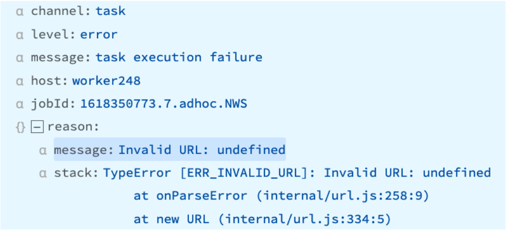
REST API Collector

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.



```
α channel: task
α level: error
α message: task execution failure
α host: worker248
α jobId: 1618350773.7.adhoc.NWS
{} ⊟ reason:
   α message: Invalid URL: undefined
   α 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: Navigate to **Settings** > **General** > **Advanced** and gradually increase the **Login Rate Limit** until this error response is no longer returned.

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.`

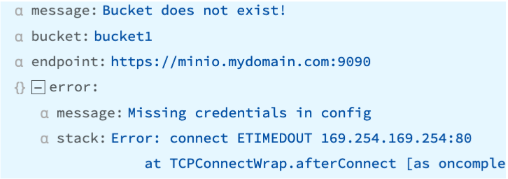
Cause: If this is persistent rather than intermittent then it could be caused by TLS negotiation failures. For example, AWS SQS currently does not support TLS 1.3. If intermittent then a network-related issue could be occurring such as DNS-related problems.

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

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 Leadr 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)

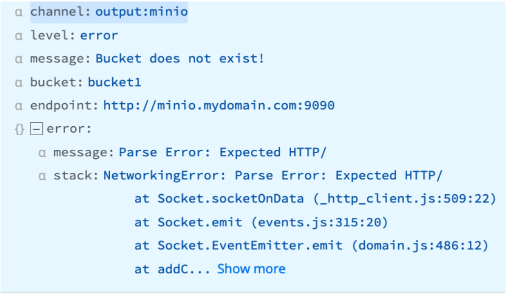
Cause: The Destination is blocking. Like the "sending is blocked" message, "begin backpressure" is logged only after 1 second of blocking. Unlike the "exerting backpressure" message, it is logged only once while backpressure is occurring (at the start), and it will always be followed by the "end backpressure" 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.



```
channel: output:minio
level: error
message: Bucket does not exist!
bucket: bucket1
endpoint: http://minio.mydomain.com:9090
error:
  message: Parse Error: Expected HTTP/
  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

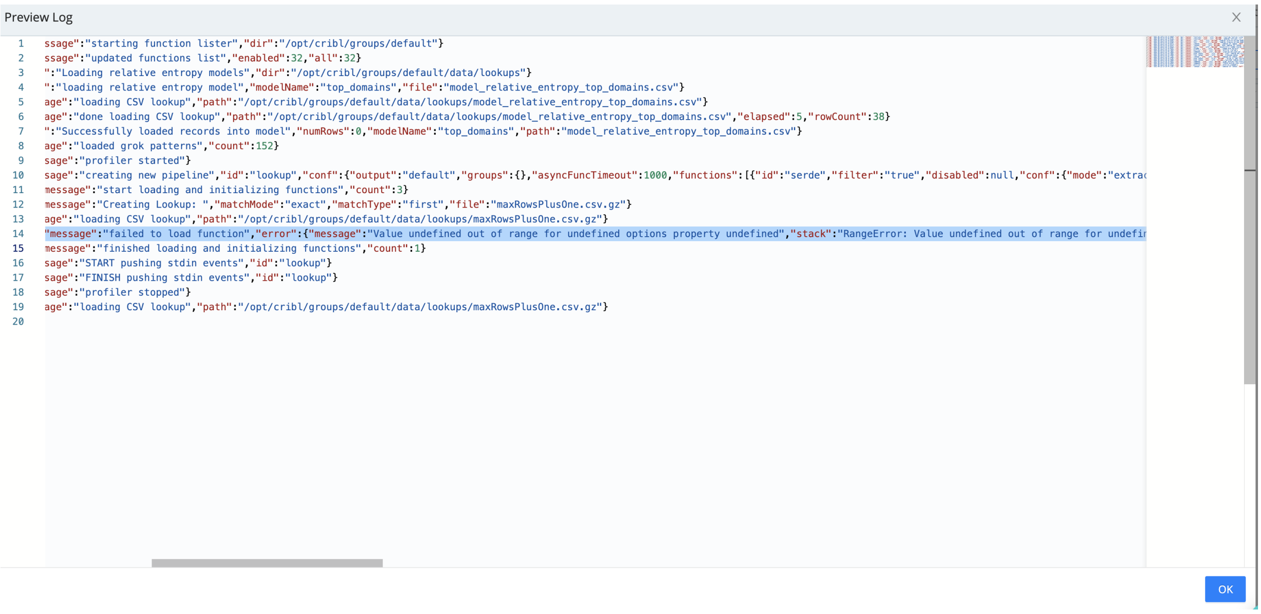
Pipelines/Functions

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.

;

13.5. 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
GitHub's recommended maximum file size of 50.00 MB
remote: error: GH001: Large files detected. You may want to try Git Large File
Storage - https://git-lfs.github.com.
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
exceeds GitHub's file size limit of 100.00 MB
```

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

;

13.6. 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`:

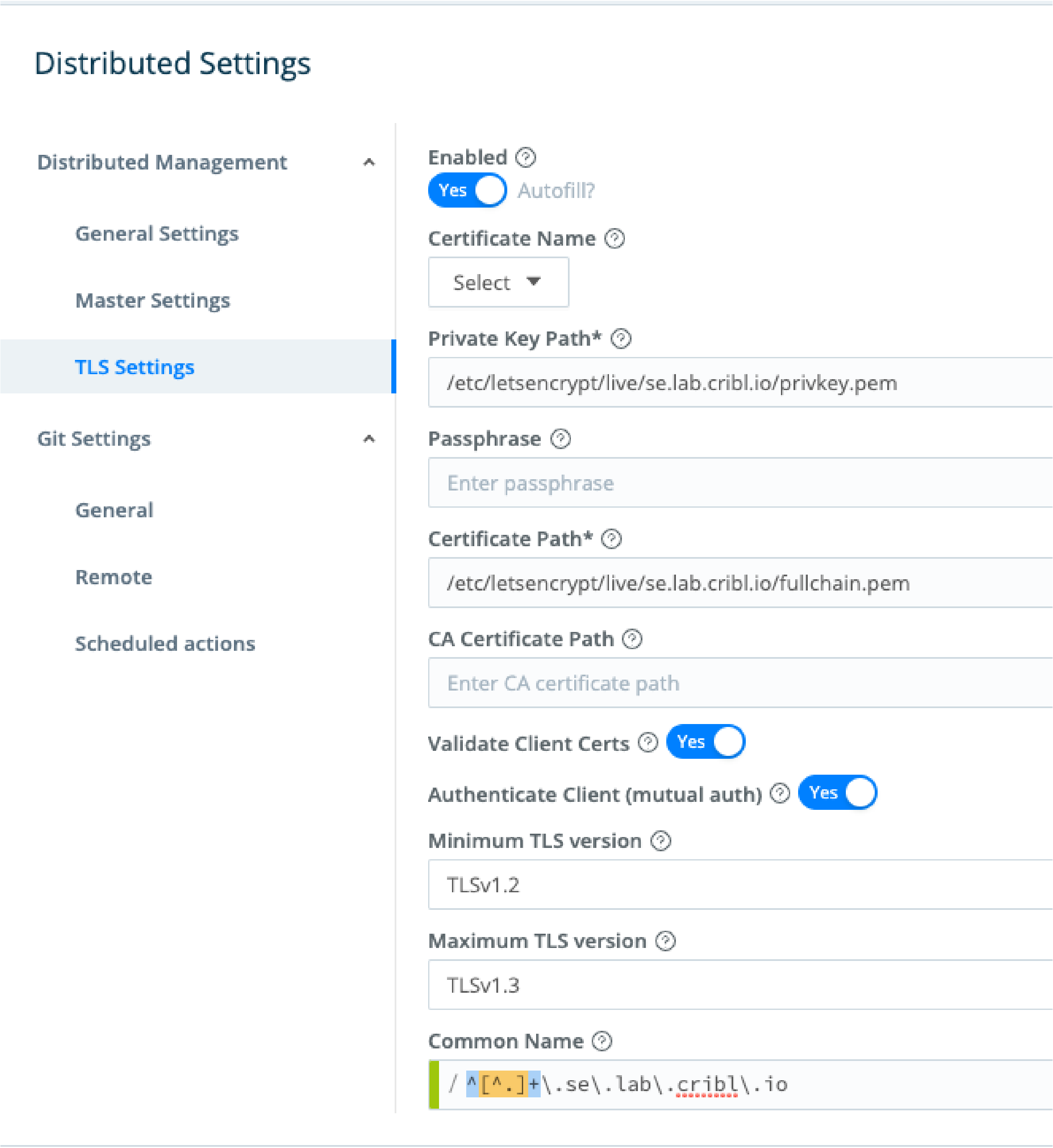
```
...
[Service]
Environment="NODE_EXTRA_CA_CERTS=/opt/cribl/local/cribl/auth/certs/ca.pem"
...
```

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.



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
'/BEGIN CERTIFICATE/,/END CERTIFICATE/p'
```

;

13.7. 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"}
```

;

14. Third-Party Software

14.1. Credits

Various components in Cribl Stream are built and enhanced with software under free or open source licenses. We thank those projects' contributors!

ag-grid-community – 19.1.2
ag-grid-react – 19.1.2
ajv – 6.9.0
ajv-errors – 1.0.1
alphanum-sort – 1.0.2
antd – 3.26.15
as-table – 1.0.36
avsc – 5.4.9
aws-sdk – 2.880.0
aws4: 1.11.0
@azure/storage-blob – 12.4.0
@azure/storage-queue – 12.4.0
bindings – 1.5.0
bl – 4.0.3
blueimp-md5 – 2.18.0
cidr-matcher – 1.0.5
clarinet – 0.12.4
classnames – 2.2.6
color-hash – 1.0.3
cron-parser – 2.15.0
d3-time – 1.1.0
d3-time-format – 2.2.3
@dabh/diagnostics – 2.0.2
date-fns – 1.29.0
diff – 3.5.0
diff2html – 3.3.1
echarts – 4.6.0
escodegen – 1.11.1
esprima – 4.0.1
express – 4.16.3
fast-array-diff – 1.0.0

fast-bitset – 1.3.2
file-saver – 1.3.8
good-fences – 0.9.1
google_protobuf – 3.15.6
@google-cloud/pubsub – 2.17.0
@google-cloud/storage – 5.8.3
@grpc/grpc-js – 1.4.4
http-proxy-agent – 3.0.0
https-proxy-agent – 4.0.0
jwt-simple – 0.5.6
kafkajs – 1.11.0
kafkajs-snappy – 1.1.0
ldapts – 1.10.0
limiter – 1.1.4
lodash – 4.17.21
lz4js – 0.2.0
maxmind – 3.1.2
Node.js – 14.18.3
node-cache – 4.2.0
node-uuid – 1.4.8
numeral – 2.0.6
pako – 1.0.10
papaparse – 5.0.0-beta.0
pbf – 3.2.1
proxy-from-env – 1.0.0
query-string – 6.1.0
react – 16.13.1
react-color – 2.19.3
react-dom – 16.13.1
react-grid-layout – 0.18.3
react-markdown – 6.0.2
react-redux – 7.2.2
react-router-dom – 5.1.2
react-sortable-hoc – 1.11.0
react-split-pane – 0.1.91
react-virtual – 2.6.1
@readme/markdown – 6.22.0
redis – 3.1.2
@reduxjs/toolkit – 1.6.1
regexpp – 2.0.0
requirejs – 2.3.6

resize-observer-polyfill – 1.5.0

rxjs – 6.5.5

saxen – 8.1.0

semver – 7.3.5

simple-git – 1.126.0

snappyjs – 0.6.0

snmp-native – 1.2.0

streamcount – 1.0.1

tar-stream – 2.1.4

timezone-support – 2.0.2

@types/d3-time – 1.0.10

url – 0.11.0

winston – 3.0.0

winston-transport – 4.4.0

xmlbuilder – 10.0.0 yaml – 1.3.2

;