



Cribl LogStream Documentation Manual

Version: v2.2

INTRODUCTION	6
About Cribl LogStream	6
Basic Concepts	7
DEPLOYMENT	9
Deployment Types	9
Single-Instance Deployment	11
Distributed Deployment	15
Bootstrap Workers from Master	23
Splunk App Deployment *	26
Sizing and Scaling	30
Config Files	33
cribl.yml	35
inputs.yml	37
outputs.yml	38
licenses.yml	40
regexes.yml	41
breakers.yml	42
mappings.yml	43
instance.yml	44
Licensing	45
User Authentication	48
Persistent Queues	51
Securing	53
Monitoring	55
Version Control	61
Upgrading	66
Diagnosing Issues	70
Uninstalling	72
WORKING WITH DATA	73
Routes	73
Pipelines	76
Event Model	79
Event Processing Order	81
Functions	83
Auto Timestamp	86
Aggregations	88

CEF Serializer	94
Clone	96
Comment	97
Drop	98
Dynamic Sampling	99
Eval	102
Flatten	105
GeoIP	107
JSON Unroll	108
Lookup	110
Mask	114
Numerify	115
Parser	116
Publish Metrics	121
Regex Extract	123
Regex Filter	125
Sampling	126
Serialize	127
Suppress	129
Tee	131
XML Unroll	133
Prometheus Publisher (beta)	136
Reverse DNS (beta)	137
Unroll	138
Sources	140
Splunk TCP	142
Splunk HEC	145
Syslog	148
Elasticsearch	150
TCP JSON	153
TCP (RAW)	156
HTTP(S)	159
Kafka	163
Kinesis Streams	166
Azure Event Hubs	168
Metrics	170
SQS	172
S3	175

SNMP Traps	179
Datagens	181
Cribl Internal	183
Collectors	185
Filesystem/NFS	189
S3	192
Script	195
Destinations	198
Output Router	201
Splunk Single Instance	203
Splunk Load Balanced	205
Splunk HEC	209
S3 Compatible Stores	211
Kinesis Streams	213
CloudWatch Logs	215
SQS	216
Filesystem/NFS	218
Elasticsearch	220
Honeycomb	222
TCP JSON	224
Syslog	226
Kafka	228
Azure Blob Storage	231
Azure Monitor Logs	233
Azure Event Hubs	235
StatsD	237
StatsD Extended	239
Graphite	241
SNMP Traps	243
InfluxDB	244
MinIO	246
Data Preview	248
Securing Data	251
Encryption	252
Decryption	256
Scripts	258
Datagens	259
CLI Reference	263

EXPRESSION REFERENCE	270
Introduction	270
Cribl Expressions	273
KNOWLEDGE	281
Regex Library	281
Event Breakers	283
Lookups Library	290
Parsers Library	292
Schema Library	294
Global Variables Library	296
USE CASES	298
Ingest-time Fields	298
Ingest-time Lookups	301
Sampling	305
Access Logs: Apache, ELB, CDN, S3 etc.	307
Firewall Logs: VPC Flow Logs, Cisco ASA etc.	310
Masking and Obfuscation	313
Regex Filtering	317
Encrypting Sensitive Data	319
KNOWN ISSUES	323
Known Issues	323
THIRD PARTY SOFTWARE	325
Credits	325

INTRODUCTION

About Cribl LogStream

Getting started with Cribl LogStream

What Is Cribl LogStream?

[Cribl LogStream](#) 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.



[Cribl LogStream](#) 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 LogStream For?

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

 Updated 10 days ago

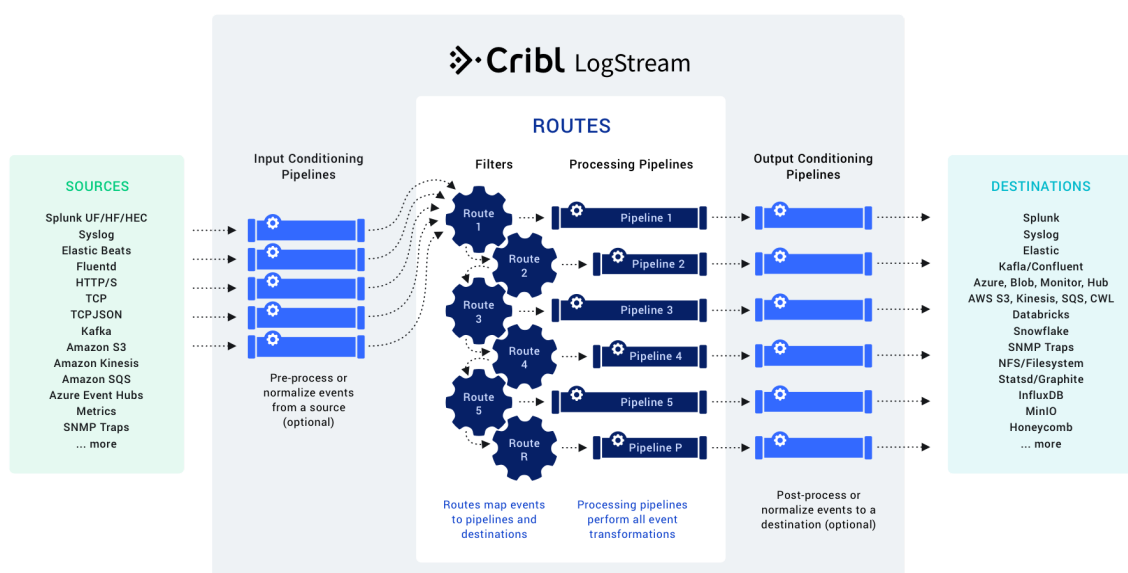
Basic Concepts

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

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



Let's zoom in on the center of the above diagram, to take a closer look at the processing and transformation options that LogStream provides internally. The basic interface concepts to work with are [Routes](#), which manage data flowing from and to [Pipelines](#), which consist of [Functions](#).



Routes

Routes evaluate incoming events against filter expressions to find the appropriate pipeline to send them to. Routes are **evaluated in order**. A Route can be associated with **only one**

Pipeline and one output. By default, a Route-Pipeline-Output tuple will consume matching events.

If the Route's `Final` flag is disabled, one or more event **clones** are sent down the associated Pipeline, while the original event continues down the rest of the 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 the Functions are executed matters. 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.



Events only move forward – toward the end of the Pipeline, and eventually out of the system. For more details, see [Pipelines](#).

Functions

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

Functions 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](#).

 Updated 4 days ago

DEPLOYMENT

Deployment Types

Deployment guide to get you started with Cribl

There are at least **two** key factors that will determine the type of Cribl LogStream 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 utilize you can deploy [Cribl App for Splunk](#). See the note below before you plan.

 **Cribl App for Splunk Deprecation Notice**

Click [here](#).

 Updated 4 days ago

Single-Instance Deployment

Getting started with Cribl LogStream on a single instance

For small-volume or light processing environments – or for test and evaluation use cases – a single instance of Cribl LogStream may be sufficient to serve all inputs, processing of events, and sending to outputs. This page outlines how to implement a single-instance deployment.

Architecture



Requirements

- **OS:**
 - Linux: RedHat, CentOS, Ubuntu, AWS Linux, Suse (64bit)
 - macOS 10.13 and 10.14
- **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).

To fulfill the above requirements using cloud-based virtual machines, see [Recommended AWS, Azure, and GCP Instance Types](#).

Network Ports

By default, Cribl LogStream needs the following ports to be available:

- UI Default: 9000
- HTTP In, Default: 10080
- Splunk to Cribl LogStream data port. Default: localhost:10000 (Cribl App for Splunk)
- | criblstream Splunk search command to Cribl LogStream. Default: localhost:10420 (Cribl App for Splunk)
- 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 .

Installing on Linux/Mac

- Install the package on your instance of choice. Download it [here](#).
- Ensure that the above ports are available.
- Un-tar in a directory of choice, e.g., /opt/ :
 - tar xvzf cribl-<version>-<build>-<arch>.tgz

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

 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 LogStream with [Sources](#) and [Destinations](#) or start creating [Routes](#) and [Pipelines](#).



Upon API port conflict, the process will retry binding for 10 minutes before exiting.

Enabling Start on Boot

Cribl LogStream ships with a CLI utility that can update your system's configuration so that LogStream can start at system boot time. Boot-start is currently supported only on Linux. Newer systems use `systemd` to start processes at boot, while older ones use `initd`.

Using `systemd`

To **enable** Cribl LogStream to start at boot time, run the following command:

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

This will install a unit file (as below) and start Cribl LogStream at boot time as user `charlize`. 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 LogStream installation:

```
[sudo] chown -R charlize $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
```

Installed systemd File

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

```
[Service]
Type=forking
User=charlize
Restart=on-failure
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
ExecStopPost='/bin/rm -f /install/path/to/cribl/pid/cribl.pid'
ExecReload=/install/path/to/cribl/bin/cribl reload
TimeoutSec=60
```

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

Using `initd`

To **enable** Cribl LogStream to start at boot time, run the following command:

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

This will install an `init.d` script in `/etc/init.d/cribl.init.d`, and start Cribl LogStream at boot time as user `charlize`. 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 LogStream installation:

```
[sudo] chown -R charlize $CRIBL_HOME
```

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

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

Do NOT run LogStream as root!

If LogStream is required to listen on ports 1–1024 it will need privileged access. On a Linux system with POSIX capabilities this can be achieved by adding the `CAP_NET_BIND_SERVICE` capability. E.g., `# setcap cap_net_bind_service=+ep $CRIBL_HOME/bin/cribl`

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](#) section for more details.

 Updated 4 days ago

Distributed Deployment

Getting started with Cribl LogStream on a distributed deployment

Distributed 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. Instances in the deployment serve all inputs, process events, and send to outputs independently. They are managed centrally by a single Master Node, which is responsible for keeping configurations in sync, and for tracking and monitoring their activity metrics.

Concepts

Single Instance – a normal Cribl LogStream instance, running by itself.

Master Node – a Cribl LogStream instance running in **master** mode, used to centrally author configurations and monitor a distributed deployment.

Worker Node – a Cribl LogStream instance running as a **managed worker**, whose configuration is fully managed by a Master Node.

Worker Group – a collection of Worker Nodes that share the same configuration.

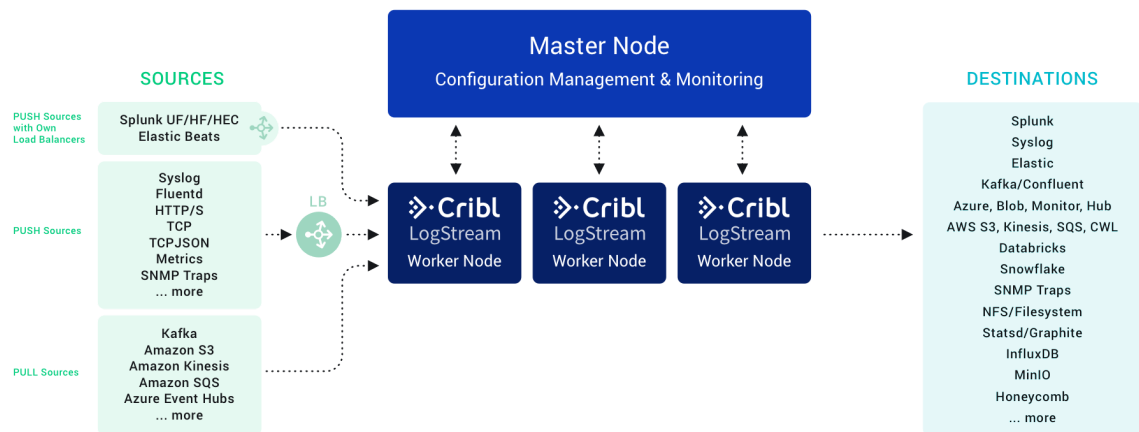
Worker Process – a process within a Single Instance or Worker Nodes that handles data inputs, processing, and output

Mapping Ruleset – an ordered list of Filters, used to map Workers to Worker Groups.



A Worker Node's local running config can be manually overridden/changed, but changes won't persist on the filesystem.

Architecture



Requirements

- See [Single-Instance Deployment](#), as the system requirements and installation procedures are identical.
- In a distributed deployment, Workers communicate to the Master Node on ports 4200 and 9000 . Ensure that Master is reachable on those ports from **all** Workers.
- You must install `git` on the Master Node. See details directly below.

Version Control with `git`

LogStream requires `git` (version 1.8.3.1 or higher) to be available locally on the host where the Master 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 Master 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 Master and Worker Nodes

1. Configuring a Master Node

Using the UI:

In **Settings | Distributed Management**, select Mode **Master**, supply the required Master settings: Address, Port, and optional settings (if used). Then click **Save** to restart.

Or, through `instance.yml` :

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: /.*/
    authToken: <auth token>
    compression: none
    connectionTimeout: 5000
    writeTimeout: 10000
```

2. Configuring a Worker Node

Using the UI:

In **Settings | Distributed Management**, select Mode **Worker**, supply the required Master settings: Address, Port, and optional settings (if used). Then click **Save** to restart.

Or, through `instance.yml`:

In `$CRIBL_HOME/local/_system/instance.yml`, under `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
```

Alternatively, you can start worker nodes with environment variables. For example:

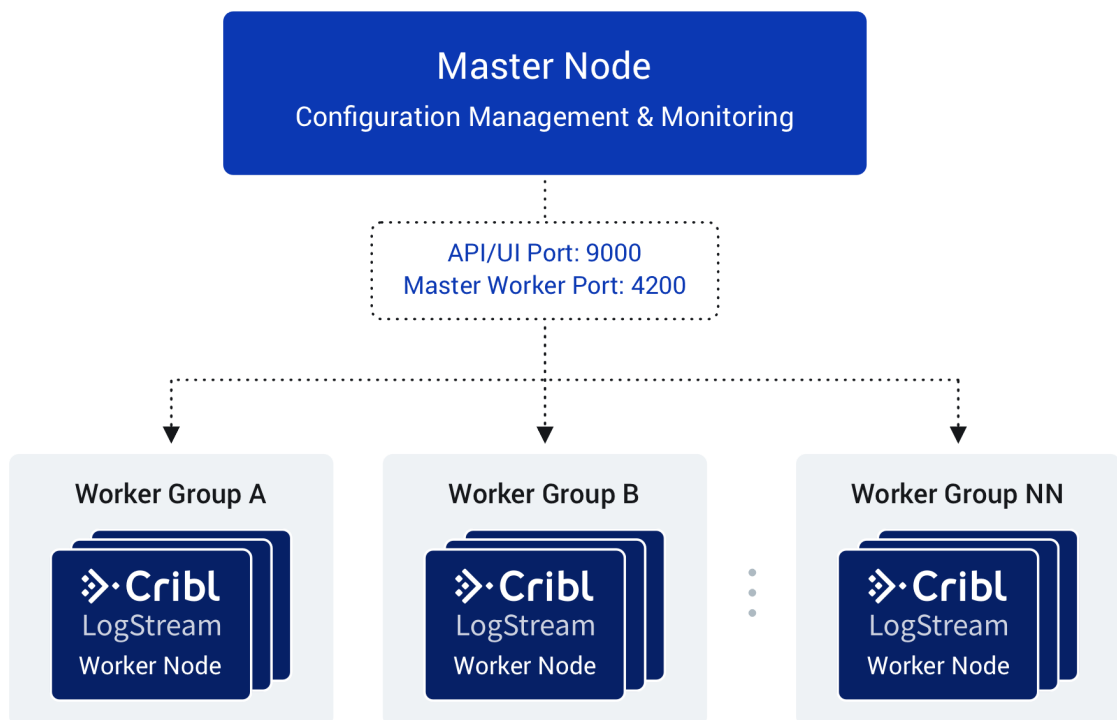
```
CRIBL_DIST_MASTER_URL=tcp://criblmaster@masterHostname:4203 ./cribl start
```

See the [environment variables](#) section for more details.

How Do Workers and Master Work Together

The Master Node has two primary roles:

1. Serves as a central location for Workers' operational metrics. The Master 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.



Workers will periodically send a heartbeat to the Master which 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 Master tracks with the connection.

When a Worker Node checks in with the Master:

- The Worker delivers its metrics to the Master.
- The Master uses the Worker's facts and Mapping Rules to map it to a Worker Group.
- The Worker Node pulls its Group's updated configuration bundle, if necessary.

Config Bundle Management

Config bundles are compressed archives of all config files and associated data that a Worker needs to operate. The Master 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 Master 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 .

Network Port Requirements (Defaults)

- UI access to Master Node: TCP 9000.
- Worker Node to Master Node: TCP 9000 (API access).
- Worker Node to Master Node: TCP 4200 (Heartbeat/Metrics).

Worker Groups

Worker Groups facilitate authoring and management of configuration settings for a particular set of Workers. To create a new Worker Group, go to the **Worker Groups** top-level menu and click **+ Add New**.

Configuring a Worker Group

Clicking on the newly created group will present you with an interface for **authoring and validating** its configuration. You can configure everything for this Group as if it were a single Cribl LogStream instance – using exactly the same visual interface for [Routes](#), [Pipelines](#), [Sources](#), [Destinations](#) and System Settings.

 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. **Only one Mapping Ruleset can be active at any one time**. A ruleset is a list of rules that evaluate Filter expressions on the information that Workers send to the Master.

The ruleset behavior is similar to [Routes](#), where the order matters and the Filter section supports full JS expressions. The ruleset matching strategy is first-match, and one Worker can belong to only one Worker Group. At least one Worker Group should be defined and present in the system.


Example: Define a rule for all hosts satisfying the condition that IP address starts with `10.10.42` , with more than 6 CPUs OR `CRIBL_HOME` environment variable contains `w0` , and

belonging to **Group420**.

- **Rule Name:** myFirstRule
- **Filter:** `(conn_ip.startsWith('10.10.42.') && cpus > 6) || env.CRIBL_HOME.match('w0')`
- **Group:** Group420

Creating a Mapping Ruleset

To create a Mapping Ruleset, start on the **Mappings** top-level menu, then click **+ Add New**.

 The **Mappings** top-level menu appears only when you have started LogStream with the **DISTRIBUTED MANAGEMENT > Mode Settings** on **Master**.

Click on the newly created item, and start adding rules by clicking on **+ Add Rule**. While working with or tuning 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 Master. To activate it, go to **Mappings** and click **Activate** on the required ruleset. You can also **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.

When an instance runs as Master, 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

The typical workflow for deploying configurations is the following:

1. Work on configs.
2. Commit (and optionally push).

3. Deploy.

Deployment is the last step after configuration changes have been saved and committed. *Deploying here means propagating updated configs to Workers.* Deploying new configurations is done at the Group level. To deploy, locate your desired Group and click on **Deploy**. Workers that belong to the group will start **pulling** updated configurations on their next check-in.



When a Worker Node pulls its first configs, the admin password will be randomized, unless specifically changed. I.e., users won't be able to log in on the Worker Node with default credentials.

Configuration Files

On the Master, a 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 Master, a group's lookup files live under: `$CRIBL_HOME/groups/<groupName>/data/` .

On the managed Worker, after configs have been pulled, lookups are extracted under:

`$CRIBL_HOME/data/` .



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.

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 LogStream node exposes a health endpoint that your Load Balancer can check to make a data/connection routing decision.

Health Check Endpoint	Healthy Response
<code>curl http://<host>:<port>/api/v1/health</code>	<code>{"status":"healthy"}</code>

Environment Variables

- `CRIBL_DIST_MASTER_URL` – URL of the Master Node. Format:
`<tls|tcp>://<authToken>@host:port?group=defaultGroup&tag=tag1&tag=tag2&tls.<tls-settings below> .`
 - `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!!
- *Deprecated variables: `CRIBL_CONFIG_LOCATION` , `CRIBL_SCRIPTS_LOCATION`*

Workers GUID

When you install and first run the software, a GUID is generated and stored in a `.dat` file located in `CRIBL_HOME/bin/` , e.g.:

```
# cat CRIBL_HOME/bin/676f6174733432.dat
{"it":1570724418,"phf":0,"guid":"48f7b21a-0c03-45e0-a699-01e0b7a1e061"}
```

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

 Updated 10 days ago

Bootstrap Workers from Master

Boot fully provisioned Workers

This feature of LogStream allows workers to completely provision themselves on initial boot directly from the master. It allows a fleet of any number of nodes to launch and be fully functional within the cluster in seconds.

How Does It Work?

A LogStream Master Node (v2.2 or higher) provides a bootstrap API endpoint, available at `/init/install-worker.sh`, which returns a shell script. You can run this shell script on any machine without LogStream installed, fully provisioning the machine as a Worker Node.

API Spec

Request Format

GET `http://<master hostname or IP>:9000/init/install-worker.sh`

Query Strings

String	Required?	Description
token	required	Master Node's shared secret (<code>authToken</code>). Can be found on the the Master Node's Distributed Settings section.
group	optional	Name of the cluster's work group. If not specified, falls back to <code>default</code> .
download_url	optional	Provide the complete URL to a Cribl installation binary. This is especially useful if the Worker Nodes don't have access to the internet to download from <code>cribl.io</code> .

Example Request

HTTP

GET `http://<master hostname or IP>:9000/init/install-worker.sh?token=79364d6e-`

Response

Shell

```
#!/bin/sh

### START CRIBL MASTER 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 -
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=

echo "Creating cribl user"
if [ $UBUNTU -eq 1 ]; then
    adduser cribl --home /home/cribl --gecos "Cribl LogStream User" --disabled-
fi
if [ $CENTOS -eq 1 ] || [ $AMAZON -eq 1 ]; then
    adduser cribl -d /home/cribl -c "Cribl LogStream User" -m
    usermod -aG wheel cribl
fi

echo "Installing LogStream"
```

Status Codes

Status Code	Reason

200 - OK	All is well. You should have received the script as a response.
403 - Forbidden	Either node is not configured as a Master, or the token provided is invalid.

Caveats

Keep the following in mind when using this feature:

- Each Worker must normally have access to the internet in order to download the Cribl LogStream 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.
- By default, Worker Nodes communicate with the Master on port 4200. Ensure that access between all Workers and the Master is open on this port.
- TLS is not enabled by default. If enabled and configured, access to this feature will be over `https` instead of `http`.
- 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. Simply use the following script, and upon launch, the Worker Node will reach out to the Master, download the script, and install and configure LogStream:

Shell

```
#!/bin/bash
curl http://<master-node-ip/host-address>:9000/init/install-worker.sh?token=<ai
```

 Updated 17 days ago

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, Cribl LogStream can be installed and configured as a Splunk app (Cribl App for Splunk) and depending on your requirements and architecture, it can run either on a Search Head or a Heavy Forwarder (strongly advised). Cribl App for Splunk **cannot** be used in a Cribl LogStream [Distributed Deployment](#) or managed by a Cribl Master Node.

Running on a Search Head (SH)

When running on a SH, Cribl LogStream is set on **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 LogStream instance's TCPJSON input on port `10420` but it's also capable of sending to any other Cribl LogStream instance listening for TCPJSON. 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 single click.

Installing the Cribl App for Splunk on a SH

- Select an instance where to install
- Ensure that ports `10000`, `10420` and `9000` are available. See Before Deploying 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 login with a Splunk **admin** role credentials.

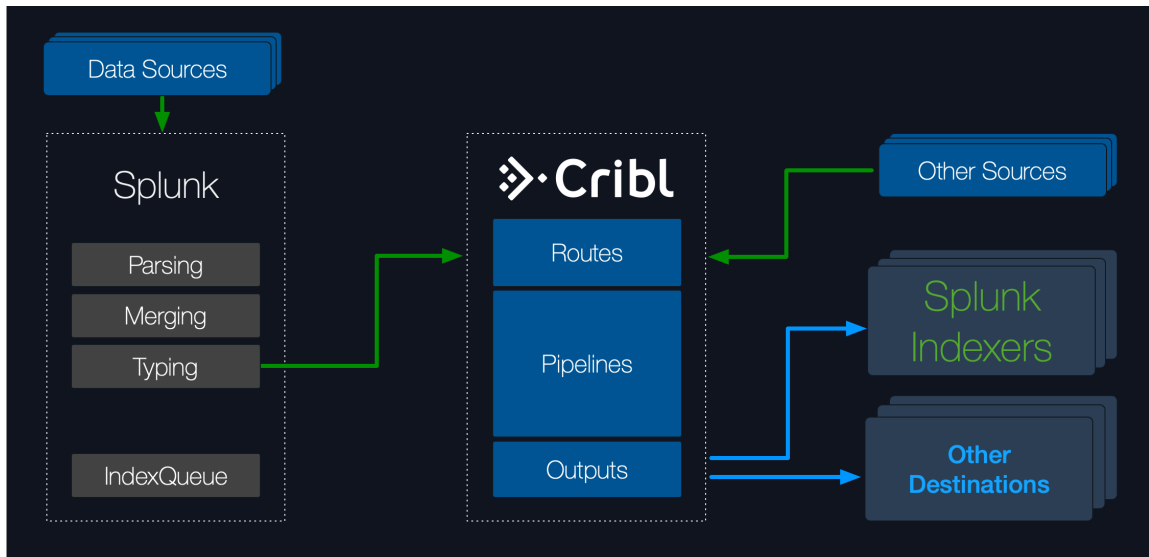
Typical Use Cases for Search Head Mode

- Working with search results in a Cribl LogStream pipeline

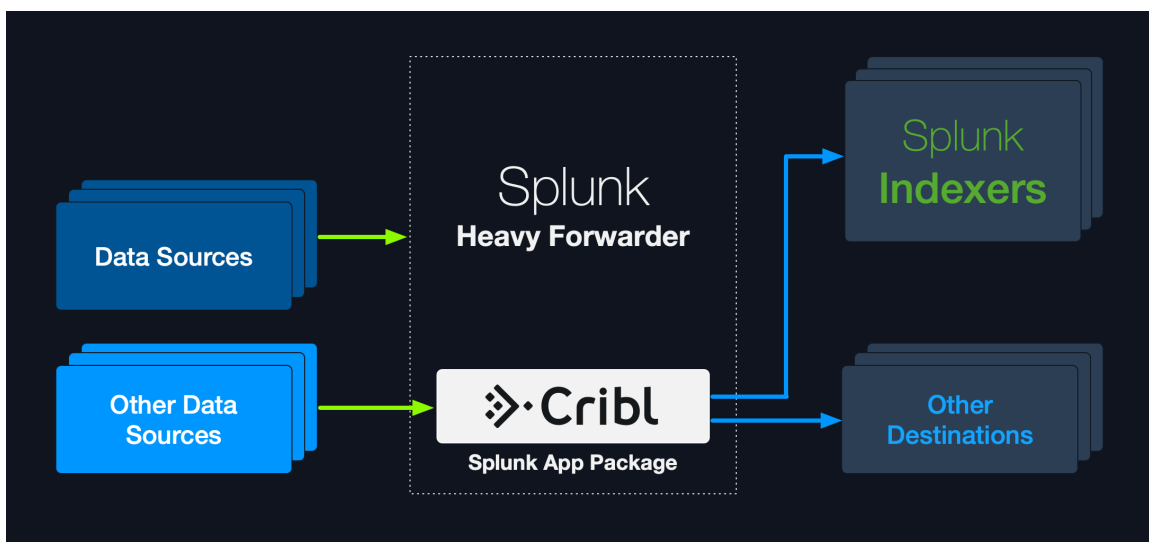
- Sending search results to any Destination supported by Cribl LogStream.

Running on a Heavy Forwarder (HF)

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



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



Installing the Cribl App for Splunk on a HF

- Select an instance where 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 in **mode-hwf**: `$SPLUNK_HOME/etc/apps/cribl/bin/cribl mode-hwf`
 - **Note:** `SPLUNK_HOME` environment variable must be defined
- Restart the Splunk instance
- Go to `https://<instance>:9000` and login with a Splunk **admin** role credentials.

Note about Splunk warnings

If you come across messages similar to below, on startup, or in logs:

```
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)
please ignore them. They are benign warns.
```

Relevant configurations in Cribl App for Splunk on a HF

When Cribl App for Splunk is installed on a HF (in `mode-hwf`), these are the **relevant sections** in configuration files that enable Splunk to send data to Cribl LogStream.

`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:typingQuei
```

`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/transforms.conf`

```
apps/cribl/default/props.conf
```

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

Configuring Cribl LogStream with a Subset of your Data

The `props.conf` stanza above will apply the above transforms to **everything**. Depending on your requirements you may want to target 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 thru Cribl.

outputs.conf

```
[tcpout]
defaultGroup = myIndexers

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

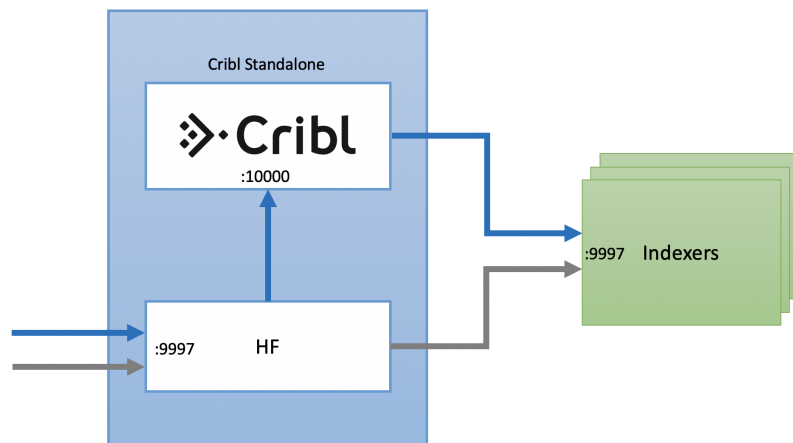
```
[tcpout:myIndexers]
server=<list of indexers>
...
```

props.conf

```
[<bluedata>]
TRANSFORMS-toCribl = route2cribl
...
```

transforms.conf

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



Configure Cribl LogStream to Send Data to Splunk Indexers

To send data from Cribl LogStream to a set of Splunk indexers, use the UI to go to **Destinations** | **Splunk Load Balanced** and enter the required information.

📄 Updated 10 days ago

Sizing and Scaling

A Cribl LogStream 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.

Scale Up

A single-instance Cribl LogStream installation can be configured to scale up and utilize as many resources on the host as required. Allocation of resources is governed through the **General Settings > Worker Processes Settings** section.


Memory (MB): Amount of memory available to each worker process, in MB. Defaults to 2048 .

Process Count: Indicates the number of worker processes to spawn. Each worker process can utilize up to 1 physical core. Negative numbers can be used to tie the number of workers relative to the number of CPUs in the system. Any setting less than 1 is interpreted as (number of CPUs available minus this setting).

 Throughout these guidelines, we assume that 1 physical core is equivalent to 2 virtual/hyperthreaded CPUs (vCPUs).

For example, assuming a Cribl LogStream system with 6 physical cores (12 vCPUs):

- If **Process Count** is set to 4 , then the system will spawn 4 processes, using up to 4 vCPUs, leaving 8 free.
- If **Process Count** is set to -2 , then the system will spawn 10 processes (12-2), using up to 10 vCPUs. This will leave 2 vCPUs free.

 LogStream incorporates guardrails that prevent spawning more processes than available vCPUs.

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

1. If data comes into a single socket, then it will be processed by a single process. So **to get the full benefits of multiple worker processes, it's important that data come in from multiple sockets.**

E.g., it's better to have 5 connections, each bringing in 200GB/day, than one doing 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.

Capacity and Performance Considerations

As with most data processing applications, Cribl LogStream's expected resource utilization will be commensurate with 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 finding and replacing a string. At the time of this writing:

- A worker process will use up to 1 physical core, or 2 vCPUs.
- Processing performance is proportional to CPU clock speed.
- All processing happens in-memory.
- Processing does not require significant disk allocation.

Estimating Requirements

Current guidance for capacity planning is: **Allocate 1 physical core for each 400GB/day of IN+OUT throughput.** So, to estimate the number of cores needed: Sum your expected input and output volume, then divide by 400GB.

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

Recommended AWS, Azure, and GCP Instance Types

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

AWS – [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)

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)

Scale Out

When data volume, processing needs, or other requirements exceed what a single instance can sustain, a Cribl LogStream deployment can span multiple nodes. This is known as a Distributed Deployment, and it can be configured and managed centrally by a single master instance. See [Distributed Deployment](#) for more details.

 Updated 10 days ago

Config Files

Understanding Configuration Paths and Files

Even though all the Routes, Pipelines and Functions can be managed from the UI, it's important to understand how the configuration works under the hood. At the time of this writing this is how configuration paths and files are laid on the filesystem.

\$CRIBL_HOME	Standalone Install: <code>/path/to/install/cribl/</code> Splunk App Install: <code>\$SPLUNK_HOME/etc/apps/cribl/</code>
---------------------	--

All paths below relative to `$CRIBL_HOME`

Default Configurations	default/cribl
Local Configurations	local/cribl
System Configuration	(default local)/cribl/cribl.yml See cribl.yml
API Configuration	(default local)/cribl/api.yml
Source Configuration	(default local)/cribl/inputs.yml See inputs.yml
Destination Configuration	(default local)/cribl/outputs.yml See outputs.yml
License Configuration	(default local)/cribl/licenses.yml
Regexes Configuration	(default local)/cribl/regexes.yml
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 Conf	(default local)/cribl/functions/<function_name>/... Each function's conf contained therein.

Configurations and Restart

- Any configuration changes resulting from UI interactions, for instance, changing the order of functions in a pipeline, or changing the order of routes, **do not require restarts**.
- All Cribl LogStream configuration file changes resulting from direct file manipulations in (bin|local|default)/cribl/... **will require restarts**.
- In the case of a Cribl App for Splunk, Splunk configurations file changes may or may not require restarts. Please check with recent 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, all changes should be done in `local`.

 Updated 10 days ago

cribl.yml

cribl.yml contains settings for configuring API and other system properties.

\$CRIBL_HOME/default/cribl/cribl.yml

```
api:
  # Address to bind to. Default: 0.0.0.0
  host: 0.0.0.0
  # Port to listen to. Default: 9000
  port: 9000
  # Flag to enable/disable UI. Default: false
  disabled : false
  # SSL Settings
  ssl:
    # SSL is enabled by default
    disabled: false
    # Path to private key
    privKeyPath: /path/to/privkey.pem
    # Path to certificate
    certPath: /path/to/cert.pem
auth:
  # Type of authentication.
  type: splunk
  host: localhost
  port: 8089
  ssl: true
workers: # worker processes, memory in MB
  count: 2
  memory: 2048
kms.local:
  # Encryption key management system settings. Default type: local.
  type: local
crypto:
  # Crypto settings.
  keyPath: $CRIBL_HOME/local/cribl/auth/keys.json
system:
  # Upgradability options: api, auto, false
  upgrade: api
  # Restart options: api, false
  restart: api
  # installType options: standalone, splunk-app
  installType: standalone
  # Flag to enable/disable intercom. Default: true
  intercom: true
license:
  accepted: true
# distributed mode: master | worker | single
distributed:
  mode: master
```

 Updated 6 months ago

inputs.yml

inputs.yml contains settings for configuring inputs into Cribl.

\$CRIBL_HOME/default/cribl/inputs.yml

```
inputs:
  # Input name
  local-splunk:
    # Input type
    type: splunk
    # Address to listen to for incoming events
    host: localhost
    # Port to listen to for incoming events
    port: 10000
  ...

secureTCPJSON:
  type: tcpjson
  disabled: false
  host: 0.0.0.0
  port: 10002
  tls:
    disabled: false
    privKeyPath: /opt/privkey.pem
    certPath: /opt/cert.pem
    requestCert: false
    rejectUnauthorized: false
  ipWhitelistRegex: /.*/
  authToken: ""
```

 Updated 7 months ago

outputs.yml

`outputs.yml` contains settings for configuring outputs from Cribl. Also see [Destinations](#) for more info.

```
$CRIBL_HOME/default/cribl/outputs.yml
```

```
outputs:
  # Default output setting
  default:
    type: default
    defaultId: local-splunk
  # Output Name
  local-splunk:
    # Output type
    type: splunk
    # Output host address to send data from
    host: localhost
    # Output port to send data from
    port: 9999
  # Output name
  myFilesystemDestination:
    # Output type
    type: filesystem
    # Final destination path. Writable by Cribl.
    destPath: /path/to/destination
    # Staging destination path. Writable by Cribl.
    stagePath: /tmp/foo
    # Partition schema for outputted files
    partitionExpr: >=
      `${host}/${sourcetype}`
    # Format of the output data
    format: json
    # The output filename prefix
    baseFileName: CriblOut
    # Compression options. None | Gzip
    compress: none
    # Maximum uncompressed output file size
    maxFileSizeMB: 32
    # Maximum amount of time to keep inactive files open.
    maxFileOpenTimeSec: 300
    # Maximum amount of time to keep inactive files open.
    maxFileIdleTimeSec: 30
    # Maximum number of files to keep open concurrently.
    maxOpenFiles: 100
  myS3Destination:
    # Output type
    type: s3
    # S3 bucket address
    bucket: s2.bucket.address.here
    # Prefix to append to files before uploading
    destPath: keyprefix
```

```
# AWS API key, if not present will fallback on env.AWS_ACCESS_KEY_ID, or tl
awsApiKey: key
# AWS Secret Key. If left blank, Cribl will fallback on env.AWS_SECRET_AC
awsSecretKey: secretkey
# Staging destination path. Writable by Cribl.
stagePath: /tmp/foo
# Partition scheme for outputted files
```

 Updated 7 months ago

licenses.yml

licenses.yml maintains a list of licenses for Cribl.

\$CRIBL_HOME/default/cribl/licenses.yml

licenses:

List of license keys

- eyJ0eXAiOiJKV1QiLCJhasdfasfasdfdasfasdfa-Abo2_ogVbR_5VKeAelZlTc5b-TKQax9R1\



Updated 7 months ago

regexes.yml

regexes.yml maintains a list of regexes. Cribl's Regex Library ships under default

\$CRIBL_HOME/default/cribl/regexes.yml

```
...
"uuid":
  lib: cribl
  description: UUID/GUID
  regex: /[0-9a-f]{8}-[0-9a-f]{4}-[1-5][0-9a-f]{3}-[89ab][0-9a-f]{3}-[0-9a-f]{12}/
  sampleData: 9a50fa34-58b1-4a67-8b8d-ea9c0ae48c8f

  eb671525-2b9e-4140-ae21-a0a8a81b506e
  tags: uuid,guid
"aws_secret_key":
  description: AWS Secret Access Key
  regex: /(?![A-Za-z0-9\/+]) [A-Za-z0-9\/+]{40}(?![A-Za-z0-9\/+]) /gm
  lib: cribl
  sampleData: wJalrXUtnFEMI/K7MDENG/bPxRfiCYEXAMPLEKEY
  tags: aws,access,key,secret
"aws_access_key":
  lib: cribl
  description: AWS Access Key ID
  regex: /(A3T[A-Z0-9]|AKIA|AGPA|AIDA|AROA|AIPA|ANPA|ANVA|ASIA) [A-Z0-9]{16}(?![A-Z0-9]|a-zA-Z0-9)/
  sampleData: >-2
  AKIAIOSFODNN7EXAMPLE
  tags: aws,access,key
"private_key":
  description: Private key block
  regex: /-----BEGIN (DSA|RSA|EC|PGP|OPENSSH) PRIVATE KEY(\\sBLOCK)?-----[\\s\\S]>
  lib: cribl
  tags: ssh,openssh,dsa,ec,rsa,private key
"slack_token":
  lib: cribl
  description: Slack Token
  regex: /xox[p|b|o|a][\\s\\S]*/g
  sampleData: xoxp-23984754863-2348975623103

  xoxa-23984754863-2348975623103

  xoxb-23984754863-2348975623103

  xoxo-23984754863-2348975623103
  tags: slack,token
...
```

 Updated 7 months ago

breakers.yml

Cribl's default Event Breaker Library is located under

`$CRIBL_HOME/default/cribl/breakers.yml`

```
$CRIBL_HOME/default/cribl/breakers.yml
```

```
...
```

```
AWS Ruleset:
```

```
  lib: cribl
```

```
  description: Event breaking rules for common AWS data sources
```

```
  tags: flowlogs,elb,alb,loadbalancer,cdn
```

```
  rules:
```

```
    - name: AWS VPC Flow
```

```
      condition: /^(\d+\s+\d+\s+eni-\w+.*(OK|NODATA|SKIPDATA)?$/.test(_raw) || :
```

```
      eventBreakerRegex: /[\n\r]+/
```

```
      timestampAnchorRegex: /(?:=\d{10}\s\d{10})/
```

```
      timestamp:
```

```
        type: format
```

```
        length: 150
```

```
        format: "%S"
```

```
      timestampTimezone: utc
```

```
      maxEventBytes: 1024
```

```
    - name: AWS ALB
```

```
      condition: /^(?:https?|h2|wss?)\s\d+--\d+--\d+.*?arn:aws:elasticloadbalanc:
```

```
      eventBreakerRegex: /[\n\r]+/
```

```
      timestampAnchorRegex: /\w+\s/
```

```
      timestamp:
```

```
        type: format
```

```
        length: 150
```

```
        format: "%Y-%m-%dT%H:%M:%S.%f%Z"
```

```
      timestampTimezone: local
```

```
      maxEventBytes: 4096
```

```
    - name: AWS ELB
```

```
      condition: /^(\d+--\d+--\d+.*?(?:\d+\.\d+\s){3})/.test(_raw) || sourcetype==
```

```
      eventBreakerRegex: /[\n\r]+/
```

```
      timestampAnchorRegex: /^/
```

```
      timestamp:
```

```
        type: format
```

```
        length: 150
```

```
        format: "%Y-%m-%dT%H:%M:%S.%f%Z"
```

```
      timestampTimezone: local
```

```
      maxEventBytes: 4096
```

```
...
```

 Updated 7 months ago

mappings.yml

Mapping ruleset configurations are located under `$CRIBL_HOME/local/cribl/mappings.yml`

`$CRIBL_HOME/default/cribl/mappings.yml`

```
...
rulesets:
  default: # ruleset name
  conf:
    functions:
      - filter: env.CRIBL_HOME.match('w0') # filter to match
        id: eval
        description: w0 # rule name/id
        final: true
        conf:
          add:
            - name: groupId
              value: "'myGroup42'" # group to map to
      - filter: env.CRIBL_HOME.match('w1')
        id: eval
        description: w1
        final: true
        conf:
          add:
            - name: groupId
              value: "'NewGroup22'"
  newruleset: # another ruleset
  conf:
    functions:
      - filter: (cpus>12 && env.CRIBL_HOME.match('w0')) || release.startsWitl
        id: eval
        description: catch all
        final: true
        conf:
          add:
            - name: groupId
              value: "'NewGroup2'"
...
```

 Updated 7 months ago

instance.yml

Instance configuration is located under `$CRIBL_HOME/local/_system/instance.yml`

`$CRIBL_HOME/local/_system/instance.yml`

```
distributed:
  # mode master | worker | single
mode: master
master:
  host: 0.0.0.0
  port: 4203
  tls:
    disabled: true
  ipWhitelistRegex: /.*/
  authToken: criblmaster
  compression: none
  connectionTimeout: 5000
  writeTimeout: 10000
group: default
envRegex: /^CRIBL_/
tags:
  - tag1
  - tag2
  - tag42
```

 Updated 7 months ago

Licensing

Every Cribl LogStream version ships with a **Free** license that allows for processing of up to 100GB/day on a single node deployment. Free licenses are restricted to one node, and **require** sending anonymized telemetry metadata to Cribl (more details [below](#)).

✓ Latest license expires on: 2020-09-15T00:00:00+00:00

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

You can add and manage licenses in **Settings > Licensing**.

License Types

Enterprise License

This is a standard license available for purchase. Contact Cribl Sales at sales@cribl.io for more information.

Sales Trial License

A license type used when preparing a POC, or a pilot, with requirements that go beyond those afforded by the Free license. Contact Cribl Sales at sales@cribl.io for more information.

Free License

A license type that allows for processing of up to 100GB/day on a single node deployment. Free licenses ship with the download package, and are restricted to one node per email address, per organization. **A free license requires sending of anonymized telemetry metadata to Cribl. This license will block inputs if sending fails after a grace period of 24 hrs.**

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 licenses – use only Enterprise 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 license expires, Cribl LogStream will fall back to Sales Trial and Free types to compute a future expiration date. However, an expired Sales Trial license cannot use Free to fall back to.



License Expiration Behavior

Upon license expiration, if there is no fallback license, Cribl LogStream will backpressure and block all incoming data.

Licensing in Distributed Deployments

In [distributed deployments](#), licenses should be configured both on the Master Node and on each of the Worker Groups. This allows for different Worker Groups to have different licensing capacities.

On Master: **Settings > Licensing**.

On Worker Group: **Worker Group > System Settings > Licensing**.

Telemetry Data

A **Free** license requires sharing of telemetry **metadata** with Cribl. This is used to help understand how to improve the product and prioritize new features. Telemetry payloads are sent to an endpoint located on `https://cdn.cribl.io/telemetry/`. For versions prior to 2.2, this endpoint is `34.220.85.61:8000`.

If you would like this feature disabled in order to deploy on your environment, please reach out to Cribl Sales at sales@cribl.io, and we will work with you to issue licenses on a case-by-case basis.

Data Shared Per Interval (roughly every minute):

- Version
- Instance's GUID
- Earliest, Latest Time
- Number of Events In, Out
- Number of Bytes In, Out
- Number of Open, Closed, Active Connections
- Number of Routes
- Number of Pipelines

 Updated 7 days ago

User Authentication

Cribl LogStream supports **local**, **Splunk**, **LDAP**, and **SSO/OpenID Connect** authentication methods.

Local Authentication

To set up local authentication, navigate to **Settings > General Settings > Authentication Settings** and select **Local**.

You can then manage users through the **Settings > Local Users** UI. All changes made to users are persisted in a file located at `$CRIBL_HOME/local/cribl/auth/users.json`.

Line format:

```
{"username":"user","first":"Elvis","last":"Bath","disabled":"false",  
"passwd":"GIBBERISH/aPNs8RtU5o9Lu2WE0j17XUA="}
```

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.

Manual Password Replacement

To manually add/change a password, replace the corresponding line's `passwd` key/value pair with a `password` key, in this format: `"password":"plainText"`. The plaintext passwords will be hashed during the next file's reload, and the `password` key will be deleted.

Explicitly Setting Worker Passwords

In a [distributed deployment](#), LogStream normally creates each Worker Group with a randomized password, separate from the admin user's password on the Master Node. This enhances security, but can lead to situations where you cannot log into a Worker Node.

To remedy these lockouts, you can explicitly push a new password to your Worker Groups. In the Master Node's UI:

1. From the top menu, select **Worker Groups**.
2. Select the Worker Group whose credentials you want to change.
3. From the **Worker Groups** submenu, select **System Settings**.
4. Select **Local Users**, then expand the user you want to update.
5. Update the **Password** field and select **Save**.

The cribl.secret File

When Cribl LogStream 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` .

Splunk Authentication

Splunk authentication is very helpful when deploying alongside Splunk, and authentication is available only to users with the Splunk `admin` role. To set up Splunk authentication:

Navigate to **Settings > General Settings > Authentication Settings** and select **Splunk**.

- Host: Splunk host address (typically a search head)
- Port: Splunk management port (defaults to `8089`)
- SSL: Set to `Yes` if enabled
- Fallback to local: Attempt local authentication if Splunk authentication is unsuccessful. Defaults to `false`.

LDAP Authentication

LDAP authentication is supported, and can be set up as follows:

Navigate to **Settings > General Settings > Authentication Settings**, and select **LDAP**.

- Secure: Enable to use a secure ldap connections (`ldaps://`). Disable for unsecure (`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.
- Search Base: Starting point to search LDAP for users, e.g., `'dc=example,dc=org'`
- Search Field: LDAP user search field, e.g., `cn` or `(cn (or uid))`
- Search Filter: LDAP search filter to apply when finding user, e.g., `(&(group=admin)(!(department=123*)))` . Optional.
- Fallback to Local: Attempt local authentication if LDAP authentication is down or mis-configured. Defaults to `No` .
- Connection Timeout (ms): Defaults to `5000` .

- **Reject Unauthorized:** Valid for secure LDAP connections, set true to reject unauthorized server certificates.

SSO/OpenID Connect Authentication

SSO/OpenID authentication is supported, and can be set up as follows:

Navigate to **Settings > General Settings > Authentication Settings** and select **OpenID Connect**.

- **Provider Name:** The name of the identity provider service. Manual entries are also allowed. **Google** and **Okta** supported natively.
- **Audience:** The Audience from provider configuration, this will be the base URL i.e.: `https://yourDomain.com:9000`
- **Client ID:** The client_id from provider configuration.
- **Client Secret:** The client_secret provider configuration.
- **Email Whitelist:** Wildcard list of emails that are allowed access.
- **Authentication URL:** The full path to the providers 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:** The full path to the providers access token url.
- **Logout URL:** The full path to the providers logout url, leave blank if the provider does not support logout or token revocation.
- **Validate Certs:** Validate certificates, set false to allow insecure self signed certificates. Defaults to **Yes** .

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 (i.e., `https://localhost:9000`). The issuer (Okta, in this example) will redirect back to this site upon authentication success or failure.

 Updated about a month ago

Persistent Queues

Persistent queuing (PQ) is a feature that helps minimize data loss if a downstream receiver is unreachable. Durability is provided by writing data to disk for the duration of the outage, and forwarding it upon recovery.

PQs are implemented on the outbound side, meaning that each [Source](#) can take advantage of a [Destination](#)'s queue.

How Does Persistent Queueing Work

Each LogStream output has an in-memory queue that helps it absorb temporary imbalances between inbound and outbound data rates. E.g., if there is an inbound burst of data, the output will store events in the queue, and output them at the rate that the receiver can sync (as opposed to blocking or dropping them). Only when this queue is full will the output impose backpressure upstream.

Backpressure behavior can be configured to either **block** or **drop**. 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 it supports backpressure, too). In drop behavior, the output will discard new events until the receiver is ready.

In some environments, the in-memory queues and their block/drop behavior are acceptable. 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 LogStream output 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.

Persistent Queue Details and Constraints

Persistent queues are:

- Available at the output side (i.e., after processing).
- Engaged only when **all of the receivers of that output** exert blocking.
- 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.

Configuring Persistent Queueing

Persistent Queueing is configured individually for each output that supports it. To enable persistent queueing, go to the output's configuration page and set the **Backpressure Behavior** control to **Persistent Queueing**. This exposes the following additional controls:

- **Max File Size:** The maximum size to store in each queue file before closing it. Enter a numeral with units of KB, MB, etc. Defaults to 1 MB .
- **Maximum Queue Size:** The maximum size amount of disk space the queue is allowed to consume. Once this limit is reached, queueing is stopped, and data will be 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` .

Minimum Free Disk Space

Sufficient disk space is required for queueing to operate properly. You configure the minimum disk space in **Settings > General Settings > Limits > Min Free Disk Space**. If available disk space falls below this threshold, LogStream will stop maintaining persistent queues, and data loss will begin.

 Updated 2 days ago

Securing

You can secure Cribl LogStream's API and UI access by configuring SSL. To do so, you can use your own private keys and certs, 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 command will generate both a self-signed cert (certified for 420 days), and an unencrypted, 2048-bit RSA private key.

In the LogStream UI, you can configure the key and cert via **Settings > Encryption Keys** and **Settings > Certificates**. Alternatively, you can edit the `local/cribl.yml` file's `api` section to directly set the `privKeyPath` and `certPath` attributes. For example:

```
cribl.yml

api:
  host: 0.0.0.0
  port: 9000
  disabled : false
  ssl:
    disabled: false
    privKeyPath: /path/to/myKey.pem
    certPath: /path/to/myCert.pem
  ...
```

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 LogStream	Default disabled	Default disabled	Default disabled
API	Worker	Master	Default disabled	Default disabled	Default disabled
Worker-to-Master	Worker	Master	Default disabled	Default disabled	Default disabled
Data	Any data sender	Cribl LogStream	Default disabled	Default disabled	Default disabled

		(Source)			
Data	Cribl LogStream (Destination)	Any data receiver	Default disabled	Default disabled	Default disabled

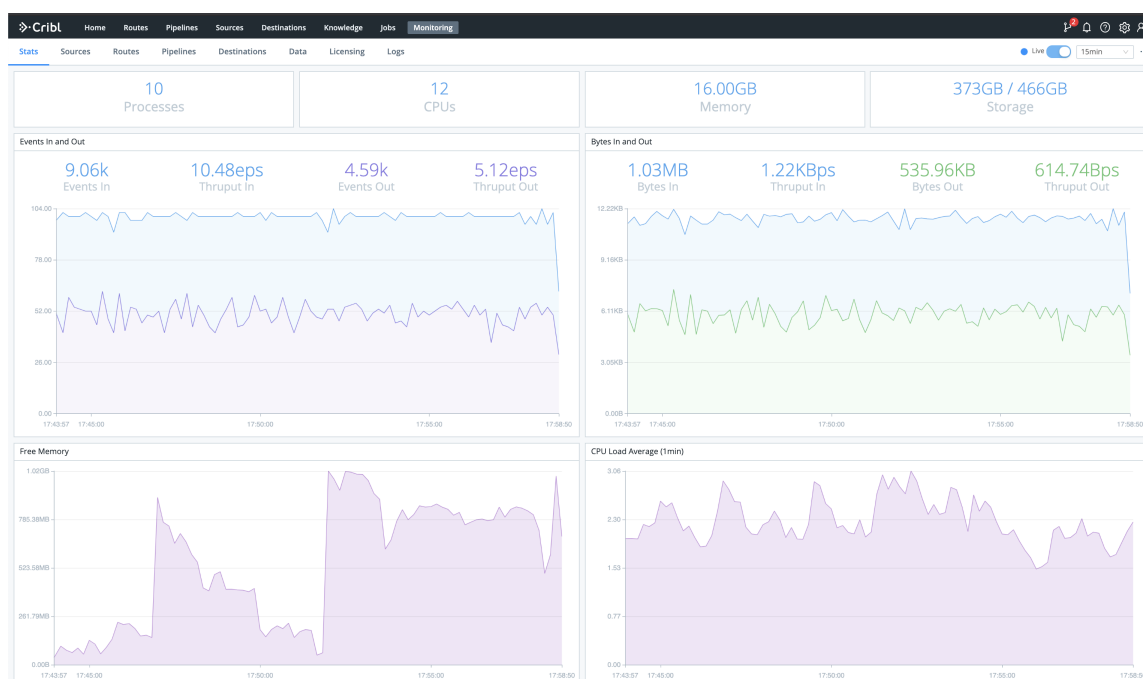
You can configure advanced, system-wide **TLS settings** for versions, cipher lists, and ECDH Curve names via **Settings > System > General Settings > Default TLS Settings**.

 Updated 24 days ago

Monitoring

To get the operational posture of a Cribl LogStream deployment, you can consult these resources:

- **Monitoring Screen:** Select **Monitoring** from the top menu. This exposes information about traffic in and out of the system. It tracks events, bytes, splits by data fields over time, and a wider range of system metrics. Coverage is limited to the previous 24 hours.



Monitoring screen

- **Internal Logs and Metrics:** Select **Logs** from the **Monitoring** submenu. LogStream's [internal logs](#) and metrics provide comprehensive information about the status of an instance, its inputs, outputs, pipelines, routes, functions, and traffic metrics.
- **Health Endpoint:** Query this endpoint on any instance to check the instance's health. (Details [below](#).)

Types of Logs

LogStream 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 in the [Diag](#) screen. Filesystem location: `$CRIBL_HOME/log/cribl.log`

- Worker Process(es) Logs – These logs are emitted by all the worker processes, and are very common in standalone instances or Worker Nodes. Filesystem location:
`$CRIBL_HOME/log/worker/N/cribl.log`
- Worker Group Logs – These logs are emitted by all processes that help a Master Node configure Worker Groups. Filesystem location:
`$CRIBL_HOME/log/group/GROUPNAME/cribl.log`

In a [distributed deployment](#), all Workers forward their metrics to the Master Node, which then consolidates them to provide a deployment-wide view.

External Monitoring

Cribl LogStream also supports forwarding internal logs and metrics to your preferred external monitoring solution. To send out internal data, go to **Sources** and enable [Cribl Internal](#). This will send all `cribl.log` logs and internal 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 Route Filters.

CriblMetrics Override

The **Disable Field Metrics** setting (in **Settings > System > General Settings > Limits**) applies only to metrics sent to the Master Node. When the **Cribl Internal** Source is enabled, LogStream ignores this **Disable Field Metrics** setting, and full-fidelity data will flow down the Routes.

Log Display and Search

LogStream exists because logs are great and wonderful things! So LogStream's **Monitoring > Logs** screen provides you with access LogStream's internal logs for both Master and Worker Nodes, from a single location. This enables you to query across all internal logs for strings of interest.

This screenshot highlights the key controls you can use (see the descriptions below):

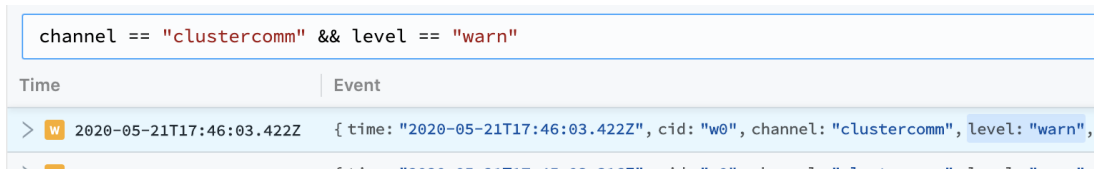
Logs screen (controls highlighted)

- Log File Selector:** Choose the Node to view. In a [Distributed Deployment](#), this list will be hierarchical, with Workers displayed inside their Master.
- Fields Selector:** Click the **Main | All | None** toggles to quickly select or deselect multiple check boxes below.
- 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.)
- Time Range Selector:** Select a standard or custom range of log data to display.
- Search Box:** To limit the displayed results, enter a JavaScript expression here. An expression must evaluate to `true` to return results. You can press **Shift+Enter** to insert a newline.

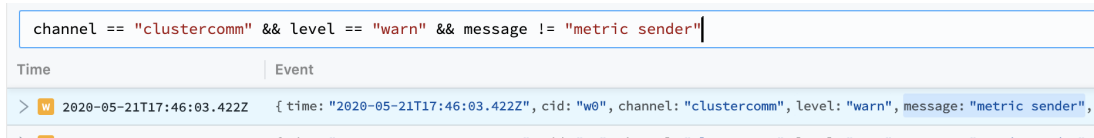
Typeahead assist is available for expression completion:

Click a field in any event to add it to a query:

Click other fields to append them to a query:

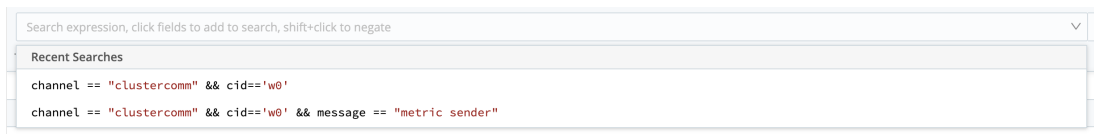


Shift+click to *negate* a field:



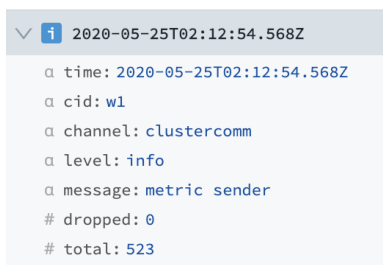
To modify the depth of information that is originally input to the Logs screen, see [Logging Settings](#).

6. Click the Search box's history arrow (right side) to retrieve recent queries:



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

Click individual log events to unwrap an expanded view of their fields:



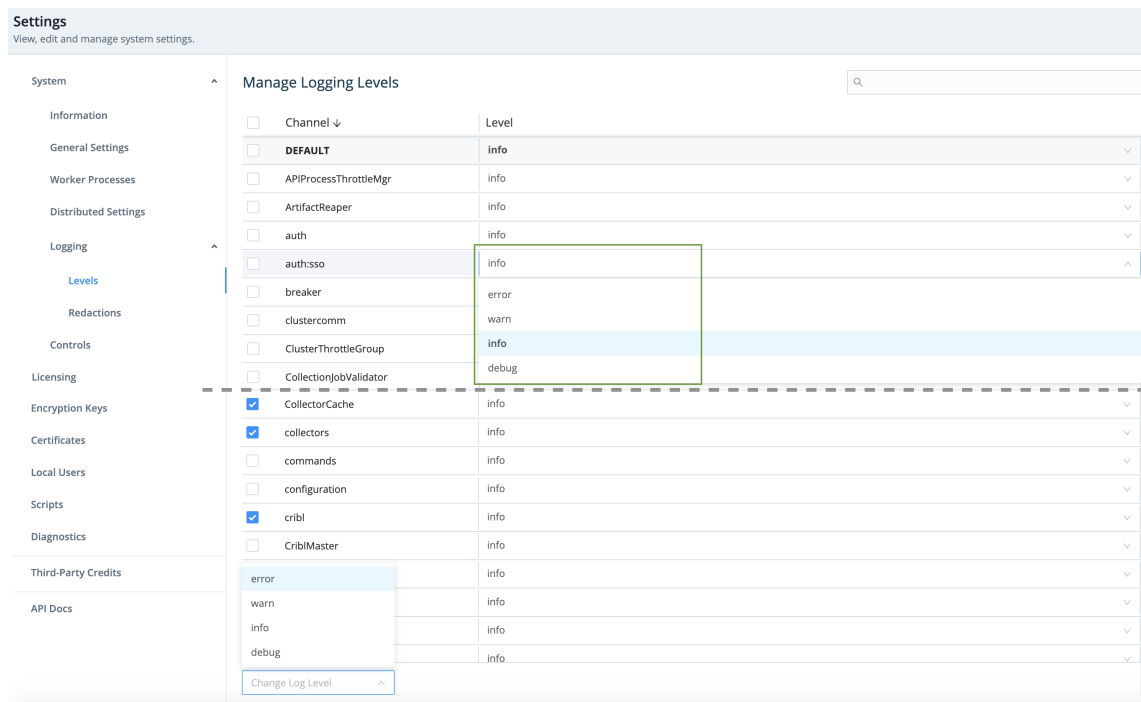
Logging Settings

Through LogStream's System Settings, you can adjust the level (verbosity) of internal logging data processed, per logging channel. You can also redact fields in customized ways.

Change Logging Levels

Select **Settings > System > Logging > Levels** to open the **Manage Logging Levels** screen. Here, 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 screen. (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 screen

Change Logging Redactions

Select **Settings > System > Logging > Redactions**: to open the **Redact Internal Log Fields** screen. Here, you can customize the redaction of sensitive, verbose, or just ugly data within LogStream's internal logs.

Redact Internal Log Fields

Custom Redact String ⓘ REDACTED

Additional Fields ⓘ field1 x field3 x

Default Fields ⓘ awsSecretKey : passphrase : authToken : azureStorageAccountKey : password : workspaceKey : team : token

Cancel Save

Redact Internal Log Fields screen

It's easiest to understand this screen's fields from bottom to top:

- **Default Fields:** LogStream always redacts these fields. You can't modify this list.

- **Additional Fields:** Type or paste in the names of other 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 LogStream's default redaction pattern, explained below.

Default Redact String

By default, LogStream transforms this screen'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** will override this default `??...??` pattern.

Health Endpoint

Each LogStream 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
<code>curl http(s)://<host>:<port>/api/v1/health</code>	<code>{"status":"healthy"}</code>

 Updated a day ago

Version Control

Cribl LogStream can be integrated with remote Git repositories to provide version control of configuration settings for standalone deployments as well as distributed deployments. (Do not confuse these options with a Git repository responsible for version control of worker configurations, located on the master node in distributed deployments.)

Git Installation

For [distributed deployments](#), `git` **must** be installed and available locally on the host running the master node.

To verify that `git` is available, run:

```
git --version
```

The minimum required version is: **1.8.3.1**

All **configuration changes must be committed before they are deployed**. The master notifies workers that a new configuration is available, and workers pull the new configuration from the master node.

Support For Remote Repositories

Git remote repositories are supported for version control of all configuration changes for Cribl from the master node, for both standalone and distributed deployments. You can configure a Standalone/Master Node with Git remote push capabilities, either from the CLI or through the UI via **Settings > Distributed Settings > Git Settings**. When a remote is configured, a **Git Push** button is displayed at the Global/Top Level config management.

Remote URI schema patterns should match this regex:

```
(?:git|ssh|ftp|file|https?|git@[-\w.]+):(\w\/)?(.*?)(\.git\/)??$ .
```

A list of supported formats can be found [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`

Connecting to GitHub with SSH

SSH keys can be set up from the CLI, or uploaded via the UI. If you have a passphrase set, this functionality is available only through the CLI.



Example: Connecting to GitHub with SSH

1. Create a new GitHub repository.
2. Add an **SSH public key** to your GitHub account.
3. In Cribl LogStream, go to **Settings > Distributed Settings > Git Settings** and fill in the remote repo URL and SSH Private Key:
Remote URL: <protocol>://git@github.com:<username>/<reponame>.git
SSH Private Key: ssh private key
Additional information: <https://help.github.com/en/github/authenticating-to-github/connecting-to-github-with-ssh>

Connecting to GitHub with a personal access token over HTTPS

(Recommended)



Example: Connecting to GitHub with a personal access token over HTTPS

1. Create a new GitHub repository.
2. Create a **personal access token** with repo privileges.
3. In Cribl LogStream, go to **Settings > Distributed Settings > Git Settings** and fill in the remote repo URL using the token in place of a password:
Remote URL: https://<username>:<token>@github.com/<username>/<reponame>.git
Additional information: <https://help.github.com/en/github/authenticating-to-github/creating-a-personal-access-token-for-the-command-line>

Reverting

Reverting to a previous commit is supported only from the CLI.

.gitignore

A `.gitignore` file specifies files that `git` should ignore. Each line specifies a pattern that ignored file paths should match. Cribl LogStream ships with a `.gitignore` file containing a number of patterns/rules, under a section of the file labeled `CRIBL SECTION`.

`.gitignore`

```
# Do NOT REMOVE CRIBL and CUSTOM header lines!
# DO NOT REMOVE rules under the CRIBL section as they may be reintroduced on up
# 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 LogStream 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 LogStream will **NOT** modify the contents of CUSTOM SECTION .

Files skipped with .gitignore

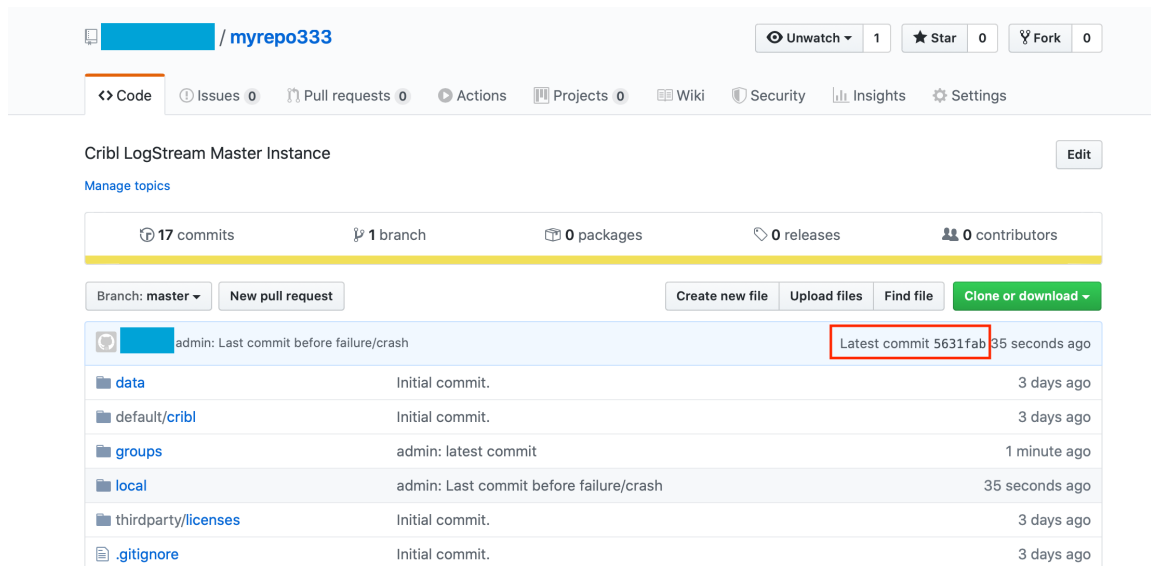
If you have files that are skipped with .gitignore, you will need to back up and restore them via other means. E.g., you can periodically copy/rsync them to a backup destination, and then restore them to their original locations after you complete the steps above.

Restoring Master From Remote Repo

If a remote repo is configured and has the latest known good Master configuration, here are the general steps to follow for restoring:

Let's assume that the entire \$CRIBL_HOME directory of the Master is corrupted, or you're starting from scratch. Let's also assume that the remote is: git@github.com:

<username>/<reponame>.git



- In a directory of choice, untar the **same** Cribl LogStream version that you're trying to restore, but do not start it.
- Ensure that you have proper access to the remote repo:

Use ls-remote to check access

```
# git ls-remote git@github.com:<username>/<reponame>.git
56331fabb4822eaec4ca0ffd008d6e9974c1e419f HEAD
5631fabb4822eaec4ca0ffd008d6e9974c1e419f refs/heads/master
```

- Change directory into `$CRIBL_HOME` and initialize git :

```
# git init
```
- Next, add/configure the remote:

```
# git remote add origin git@github.com:<username>/<reponame>.git
```
- Lastly, setup local to exactly match the remote branch:

```
# git fetch origin
# git reset --hard origin/master
```

To confirm that the commits match, run this command while in `$CRIBL_HOME` . Note commit hashes.

Confirm with git show

```
# git show --abbrev-commit
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

.....
```

That last step above pulls in all the latest configs from the remote repo, and you should be able to start the Master as normal. Once up and running, Workers should start checking in after about 60s.

 Updated 11 days ago

Upgrading

This page outlines how to upgrade Cribl LogStream's [Single-Instance](#) or [Distributed Deployment](#) packages along one of the following supported upgrade paths:

- v2.x ==> v2.x
- v1.7.x/v2.0.x ==> v2.x.x
- v1.6.x or below ==> v1.7.x ==> v2.x.x



LogStream does **not** support direct upgrades from a Beta to a GA version. To get the GA version running, you must perform a new install.

Standalone/Single-Instance

This path requires upgrading only the single/standalone node:

1. Stop Cribl LogStream.
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 LogStream.

Distributed Deployment

For a distributed deployment, the order of upgrade is: Upgrade first the Master Node, then upgrade the Worker Nodes, then commit and deploy the changes on the Master.

Upgrade the Master 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 LogStream.
 - Optional but recommended: Back up the entire `$CRIBL_HOME` directory.

- Optional: Check that the Worker Nodes are still functioning as expected. In absence of the Master Node, they should continue to work with their last deployed configurations.

3. Uncompress the new LogStream version on top of the old one.

4. Restart LogStream and log back in.

5. Wait for all the Worker Nodes to report to the Master, and ensure that they are correctly reporting the last committed configuration version.

i Workers' UI will not be available until the Worker version has been upgraded to match the version on the Master. Errors like those below will appear until the Worker nodes are upgraded.

Manage Worker Nodes
View and manage Workers.

4

1

4

0

1

1

WORKERS

GROUPS

ALIVE

UNHEALTHY

SOFTWARE VERSIONS

CONF VERSIONS

C

Q

<input type="checkbox"/>	#	GUID	Host	Status	Group	CPUs	RAM	Last Time	Start Time	Config Ve...	Version
<input type="checkbox"/>	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...

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

Restart

Worker Details

firstMsgTime: 1591388339455

group: default

id: bee650c4-97e5-4f59-ba6e-9da4b9018be1

info: 10 items...

lastMsgTime: 1591389368748

status: healthy

workerProcesses: 2

<input type="checkbox"/>	> 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...
<input type="checkbox"/>	> 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...
<input type="checkbox"/>	> 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 Node version mismatch

Upgrade the Worker Nodes

These are the same basic steps as when upgrading a [Standalone Instance](#), above:

1. Stop Cribl LogStream on each Worker Node.
2. Uncompress the new version on top of the old one.
3. Restart LogStream.

Commit and Deploy Changes on the Master Node

1. Ensure that newly upgraded Worker Nodes report to the Master with their new software version.

- 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

<input type="checkbox"/>	#	GUID	Host	Status	Group	CPU	RAM	Last Time	Start Time	Config Ve...	Version
<input type="checkbox"/>	1	b396df9e-6e49-488d-8479-0b501c9d...	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-8479-0b501c9d...

Host: 1e9a7eeb5765

Status: ✓ alive

Group: default

CPU: 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

Restart

Worker Details

firstMsgTime: 159138998446

group: default

id: b396df9e-6e49-488d-8479-0b501c9d...

info: 11 items...

lastMsgTime: 1591391726555

status: healthy

workerProcesses: 4

| ☐ | 2 | f83a2153-7466-45d9-b2fe-aab2b6edce04 | 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

CPU: 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

Restart

Worker Details

firstMsgTime: 1591390066888

group: default

id: f83a2153-7466-45d9-b2fe-aab2b6edce04

info: 11 items...

lastMsgTime: 1591391734227

status: healthy

workerProcesses: 4

Post-2.1.4 upgrade to 2.2

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:

- Stop Splunk.
- 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.

- 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.

 Updated 4 days ago

Diagnosing Issues

To help diagnose LogStream 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 LogStream was configured and operating at that time.

What's in the Diagnostic Bundle

The following directories (and their contents) off of `$CRIBL_HOME` are included:

- `/default/*`
- `/local/*`
- `/log/*`
- `/groups/*`
- `/groups/*`
- `state/jobs` - includes only latest 10 task of latest 10 jobs.

Creating and Exporting a Diagnostic Bundle

Users can create and share bundles 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 Case number to share the bundle with Cribl Support.

Using the UI

To create a bundle, go to **Settings > 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**, supply your Case number, then click **Export**.

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.

`diag` command CLI

```
# $CRIBL_HOME/bin/cribl diag
Usage: [sub-command] [options] [args]
```

Commands:

```
get      - List existing Cribl LogStream diagnostic bundles
create   - Creates diagnostic bundle for Cribl LogStream
send     - Send LogStream diagnostic bundle to Cribl Support, args:
            -c <caseNumber> - Cribl Case Number
            [-p <path>]      - Diagnostic bundle path (if empty, then new bundle will be
```

Creating a diagnostic bundle

```
# $CRIBL_HOME/bin/cribl diag create
Created Cribl LogStream diagnostic bundle at /opt/cribl/diag/cribl-logstream-<|
```

Creating and sending a diagnostic bundle

```
# $CRIBL_HOME/bin/cribl diag send -c 420420
Sent LogStream diagnostic bundle to Cribl Support
```

Sending a previously created diagnostic bundle

```
# $CRIBL_HOME/bin/cribl diag send -p /opt/cribl/diag/cribl-logstream-<hostname:
Sent LogStream diagnostic bundle to Cribl Support
```

 Updated 6 days ago

Uninstalling

Uninstalling the Standalone version

- Stop Cribl LogStream (stopping main process)
- Backup necessary configurations/data
- Remove the directory where Cribl LogStream is installed

Uninstalling the Splunk App version

- Stop Splunk
- Backup necessary configurations/data
- Remove the Cribl App in \$SPLUNK_HOME/etc/apps
- Remove the Cribl module in \$SPLUNK_HOME/etc/modules/cribl (some versions)



Updated 5 months ago

WORKING WITH DATA

Routes

What Are Routes

Before incoming events are transformed by a processing pipeline, Cribl LogStream uses a set of filters to first select a **subset** to deliver to the correct pipeline. This process is done via Routes.

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 (e.g., `source=='foo.log' && fieldA=='bar'`, `true`, etc.) that are configured with each route.

i There can be multiple routes in the system, but a route can be associated with only **one** pipeline.

Routes are evaluated in their display order, top->down.

Data ▾	Routes	Pipelines	Knowledge	System Settings			
					+ Add Route		
↕	#	Route	Filter ▾	Pipeline/Output	Events	Show All ▾	
	1	Collection Processing Rou...	<code>collection=='demo_collector' && sourcetype=='pan:traffic'</code>	passthru	0.000%		...
	2	Palo Alto Firewall Traffic	<code>sourcetype=='pan:traffic'</code>	firewall_geolp_enrich router:pan-firewall-external	0.847%		...
	3	Archival	<code>true</code>	passthru s3:s3	24.682%		...
	4	Trim BigJSON	<code>index=='cribl' && sourcetype=='lambda'</code>	trim_json	2.576%		...
	5	Logs to Metrics	<code>C.vars.accessCombined()</code>	logs_to_metrics router:statsd	8.686%		...
	6	Enrich	<code>C.vars.accessCombined()</code>	enrich	17.372%		...
	7	KV to JSON	<code>index=='cribl' && sourcetype=='business_event'</code>	kv2json	0.731%		...
	8	JWT Decode	<code>index=='cribl' && sourcetype=='authfailed'</code>	jwt_to_json	0.000%		...
	9	Sensitive Data	<code>index=='cribl' && sourcetype=='business_event'</code>	masking	0.731%		...
	10	Suppression	<code>C.vars.criblIndex(sourcetype=='nagios')</code>	suppress	2.583%		...
	11	Sample and Filter	<code>C.vars.accessCombined()</code>	sample_and_filter	17.109%		...
	12	Main Route	<code>true</code>	main	24.682%		...

In this example, incoming events will be evaluated first against the route named **Route**, then against **Sensitive Data**, then against **Logs to Metrics**, 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.

To apply a route before another, simply drag it vertically. In addition, you can turn routes **On/Off** inline as necessary.

Output Destination

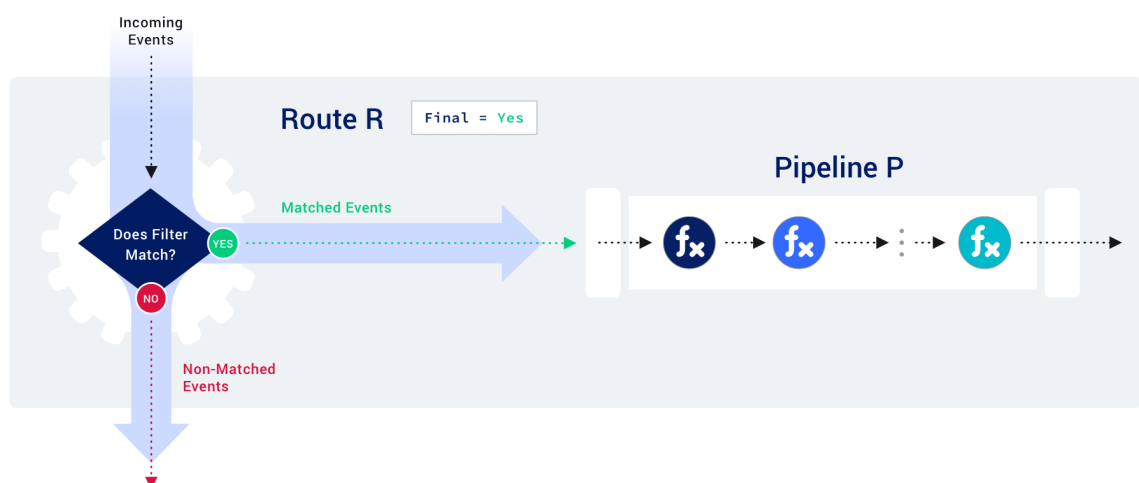
Routes can be configured with an output [destination](#) that denotes where to send events after they're processed by the pipeline.

The Final Toggle

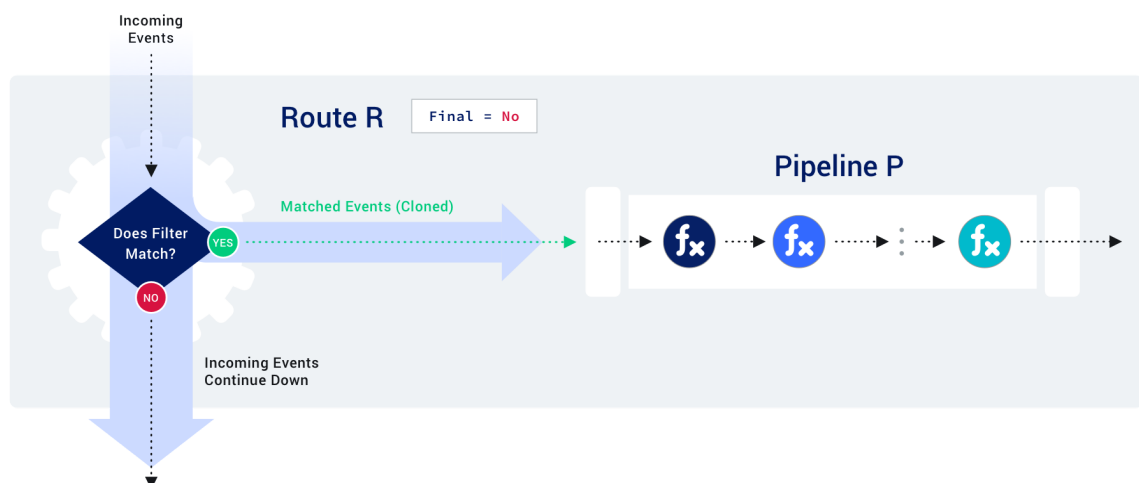
When an event that enters the system and matches a route-pipeline pair, usually it will either be:

- 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.



This is very useful in cases where the same set of events needs to be processed differently and delivered to different destinations. Each clone can be decorated with key/value pairs as necessary.

Final Flag and Cloning Considerations

Depending on your **cloning** needs, you may want to follow a **most specific first** or **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 may make sense to do that upfront, and follow it with a route that consumes those clones immediately after.

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, and they are a UI artifact only – i.e., while routes are in a group, those routes maintain their global position order.

Routing with Output Router


[Output Routers](#) are another way to route data. They are meta-destinations, in that they allow actual destination selection based on rules. Rules are evaluated in order, top->down, with the first match being the winner.

 Updated 5 days ago

Pipelines


What Are Pipelines

After your data has been matched by a [route](#), it gets delivered to a pipeline. A pipeline is a list of [functions](#) that work on the data. As with routes, the order in which the functions are listed matters.

 Functions in a pipeline are evaluated in order, top->down.

How Do Pipelines Work

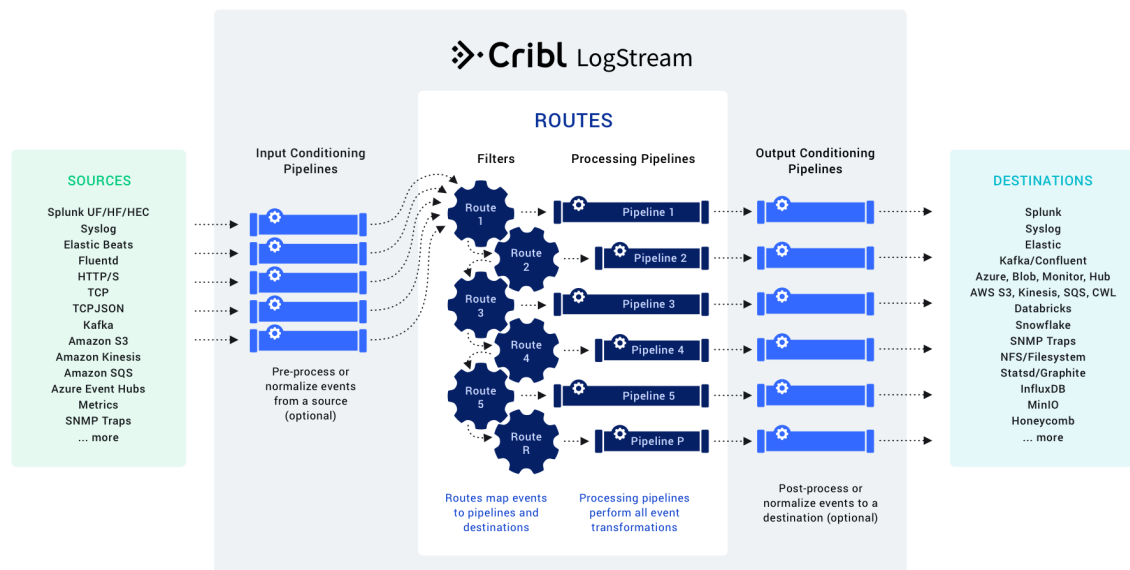
Events are always delivered to the beginning of a pipeline via a route, and they are processed by each function, in order. A pipeline will always move events in the direction that points outside of the system. This is on purpose, so as to keep the design simple and avoid potential loops.

Data ▾	Routes	Pipelines	Knowledge	System Settings
<input type="text" value="Search pipelines"/> Show All All Processing Conditioning + Add Pipeline				
<input type="checkbox"/>  Pipeline Name ↑	Route / Input	Functions	Output	Stats
<input type="checkbox"/> syslog-conditioning	syslog	2	None	IN 14.52k OUT 14.52k ERR 0 ...
<input type="checkbox"/> suppress	Suppression	3	default	IN 10.92k OUT 492.00 ERR 0 ...
<input type="checkbox"/> sample_and_filter	Sample and Filter	5	default	IN 72.69k OUT 18.55k ERR 0 ...
<input type="checkbox"/> passthru	Collection Pr... sing Route Archival	1	default s3	IN 105.47k OUT 105.47k ERR 0 ...
<input type="checkbox"/> palo_alto_traffic	None	8	None	IN 0 OUT 0 ERR 0 ...
<input type="checkbox"/> masking	Sensitive Data	2	default	IN 3.45k OUT 3.45k ERR 0 ...
<input type="checkbox"/> main	Main Route	1	default	IN 105.47k OUT 105.47k ERR 0 ...
<input type="checkbox"/> logs_to_metrics	Logs to Metrics	8	statsd	IN 36.35k OUT 30.90k ERR 0 ...
<input type="checkbox"/> kv2json	KV to JSON	3	default	IN 3.45k OUT 3.45k ERR 0 ...
<input type="checkbox"/> jwt_to_json	JWT Decode	3	default	IN 0 OUT 0 ERR 0 ...
<input type="checkbox"/> firewall_geoip_enrich	Palo Alto Firewall Traffic	16	pan-firewall-external	IN 3.60k OUT 3.60k ERR 0 ...
<input type="checkbox"/> enrich	Enrich	16	default	IN 72.69k OUT 72.69k ERR 0 ...

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.



Input Conditioning Pipelines

These are pipelines that are attached to a Source to condition (normalize) the events **before** they're delivered to a Processing Pipeline. They are optional.

Typical use cases are event formatting, or applying functions to *all* events of an input. (E.g., extract a `message` field before pushing events to various processing pipelines.) You configure these conditioning pipelines on individual [Sources](#).

Processing Pipelines

These are "normal" event processing pipelines.

Output Conditioning Pipelines

These pipelines are attached to a Destination to condition (normalize) the events **before** they're sent out. Typical use cases are applying functions that transform or shape events per receiver requirements. (E.g., ensure that a `_time` field exists for all events bound to a Splunk receiver.) You configure these conditioning pipelines on individual [Destinations](#).

Considerations

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**–**f2**, and if **f1** operates on `source 'foo'` and **f2** that operates on `source 'bar'`, it might make sense to apply `source=='foo'` versus `source=='bar'` filters on these two filters, respectively.

 Updated 4 days ago

Event Model

All data processing in Cribl LogStream 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 LogStream **event** is as follows:

Cribl LogStream 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>",
  "...": "..."
}
```

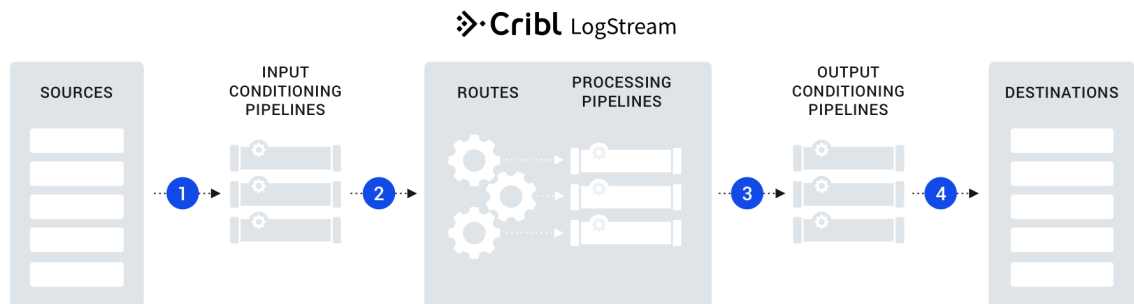
- Fields that start with a double-underscore are known as **internal fields**, and each Source can add one or many to each event. For example, Syslog adds both a `__inputId` and a `__srcIpPort` field. Internal fields are used only within Cribl LogStream, and are not passed down to Destinations.
- Upon 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 four options:

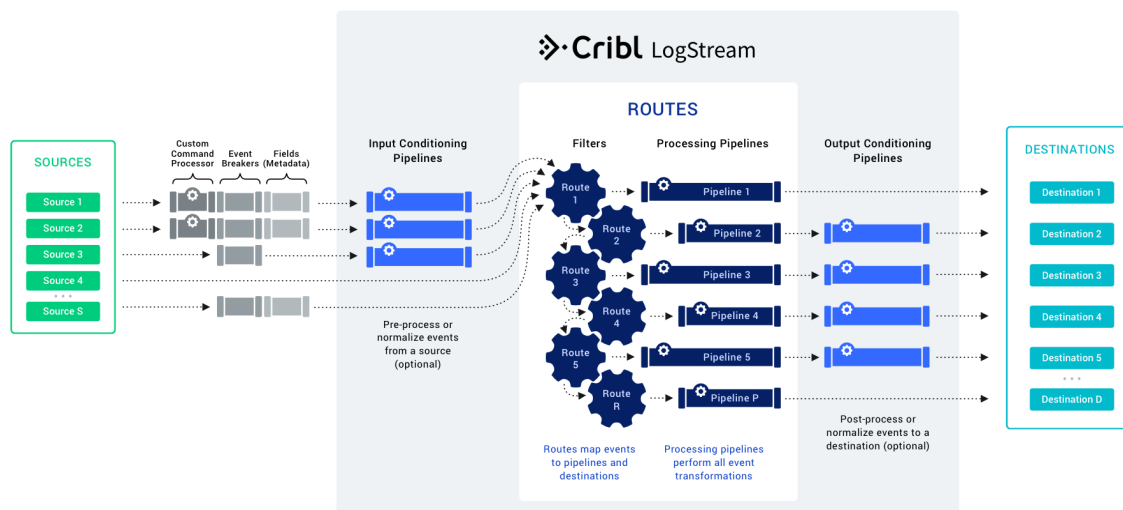
- **1. As they come in** – Capture events right after they're delivered by the respective Input.
- **2. After input pipeline** – Capture events right after the Input Conditioning Pipeline, before they go down the Routes.
- **3. After processing pipeline** – Capture events right after the Processing Pipeline that actually handled them.
- **4. After output pipeline** – Capture events right after the Output Conditioning Pipeline, before they go out to the configured Destination.



 Updated 10 days ago

Event Processing Order

Viewed according to the expanded schematic below, all events in the Cribl LogStream ecosystem are processed linearly, from left to right.



LogStream in great detail

Here are the stages of event processing:

1. **Sources**: Data arrives from your choice of external providers. (LogStream supports Splunk, HTTP/S, Elastic Beats, Amazon Kinesis/S3/SQS, Kafka, TCP raw or JSON, and many others.)
2. Custom command processor: 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`, and will process its output via `stdout`.
3. **Event Breakers** can, optionally, break up incoming bytestreams into discrete events.
4. Fields/metadata: Optionally, you can add these enrichments to each incoming event. You add fields by specifying key/value pairs, per Source, in a format similar to LogStream's **Eval** function. Each key defines a field name, and each value is a JavaScript expression (or constant) used to compute the field's value.
5. Input conditioning pipeline: Optionally, you can use a single pipeline to pre-process (normalize) data from this input before the data reaches the Routes.
6. **Routes** map incoming events to Processing Pipelines and Destinations. A Route can accept data from multiple Sources, but each Route can be associated with only one Pipeline and one Destination.

7. Processing [Pipelines](#) perform all event transformations. Within a Pipeline, you define these transformations as a linear series of [Functions](#). A function is an atomic piece of JavaScript code invoked on each event.
8. Output conditioning pipeline: Optionally, you can append a pipeline a to post-process (normalize) data from each Processing Pipeline before the data reaches its Destination.
9. [Destinations](#): Each Route/Pipeline combination forwards processed data to your choice of streaming or storage destination. (LogStream 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.

 Updated 10 days ago

Functions

What Are 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.

The screenshot shows the Splunk Pipelines configuration page for a pipeline named 'sample_and_filter'. The interface includes a top navigation bar with tabs for Data, Routes, Pipelines (selected), Knowledge, and System Settings. Below the navigation bar, there's a header for the pipeline with a '+ Add Function' button and a settings icon. The pipeline is attached to a route named 'Sample and Filter'. A summary bar shows 'IN 73.15k', 'OUT 18.72k', and 'ERR 0'. The main table lists the functions in the pipeline:

#	Function	Filter	On/Off
1	Comment	Extract the HTTP Status and Sample successes.	...
2	Regex Extract	source=='smart_sample'	On
3	Sampling	source=='smart_sample'	On
Filter Configuration for Function 3:			
Filter: source=='smart_sample'			
Description: Enter a description			
Final: No			
Sampling Rules:			
Filter	Sampling Rate		
__status == 200	5	X	
+ Add Rule			
4	Comment	Drop purchase requests	...
5	Drop	source=='filter' && !(/\"S+\"s\"S+action=purchase/i.test(_raw))	On

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:

- `off` (**default**): means that matching events processed by this function will be passed down to the next function.
- `on` : means that this function is the last one that will be applied to matching events. All functions further down the pipeline will be skipped.

Out-of-the-Box Functions

Cribl LogStream 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.

Custom Functions

Cribl LogStream does not yet support custom functions.

What Functions to Use When

- Add, remove, update fields
[Eval](#), [Lookup](#)
- Find & Replace, including basic `sed`-like, obfuscate, redact, hash etc.
[Mask](#)
- 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 / change format (e.g., convert JSON to CSV)
[Serialize](#)

- Convert JSON arrays or XML elements into 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 in metrics format
[Publish Metrics](#), [Prometheus Publisher \(beta\)](#)
- Resolve hostname from IP address
[Reverse DNS \(beta\)](#)

 Updated 4 days ago

Auto Timestamp

Description

The `Auto Timestamp` function extracts time to a destination field, given a source field in the event.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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: Timezone used to parse timestamps that lack timezone info. Defaults to `Local`.

Advanced Settings

Time Expression: Expression to use to format extracted time. Current time, as a JavaScript Date object, is in global `time`. Defaults to `time.getTime() / 1000`.

Max Timestamp Scan Depth: Maximum string length at which to look for a timestamp.

Additional Timestamps: Add Regex/Strptime pairs to extract additional timestamp formats.

- **Regex:** Regex, with first capturing group matching the timestamp.
- **Strptime Format:** Timestamp in strptime format.

Format Reference:

https://github.com/d3/d3-time-format#locale_format

`%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 (%).

Directives marked with an asterisk (*) may be affected by the locale definition.

 Updated 12 days ago

Aggregations

Description

The `Aggregations` function performs aggregate statistics on event data.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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]$`.

Aggregate(s): 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.

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: Determines if the aggregations should be retained for cumulative aggregations, or reset to 0, when flushing out an aggregation table event. Defaults to `No`.

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 passthrough 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 events to include in any given aggregation event. Defaults to unlimited.

Aggregation Memory Limit: The memory usage limit to impose upon aggregations. Defaults to unlimited (i.e., the amount of memory available in the system).

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.

`max(expr:FieldExpression)` : Returns the maximum value of the 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.

`stddev(expr:FieldExpression)` : Returns the sample standard deviation of the values of the

parameter.

`stddevp(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.

`variance(expr:FieldExpression)` : Returns the sample variance of the values of the parameter.

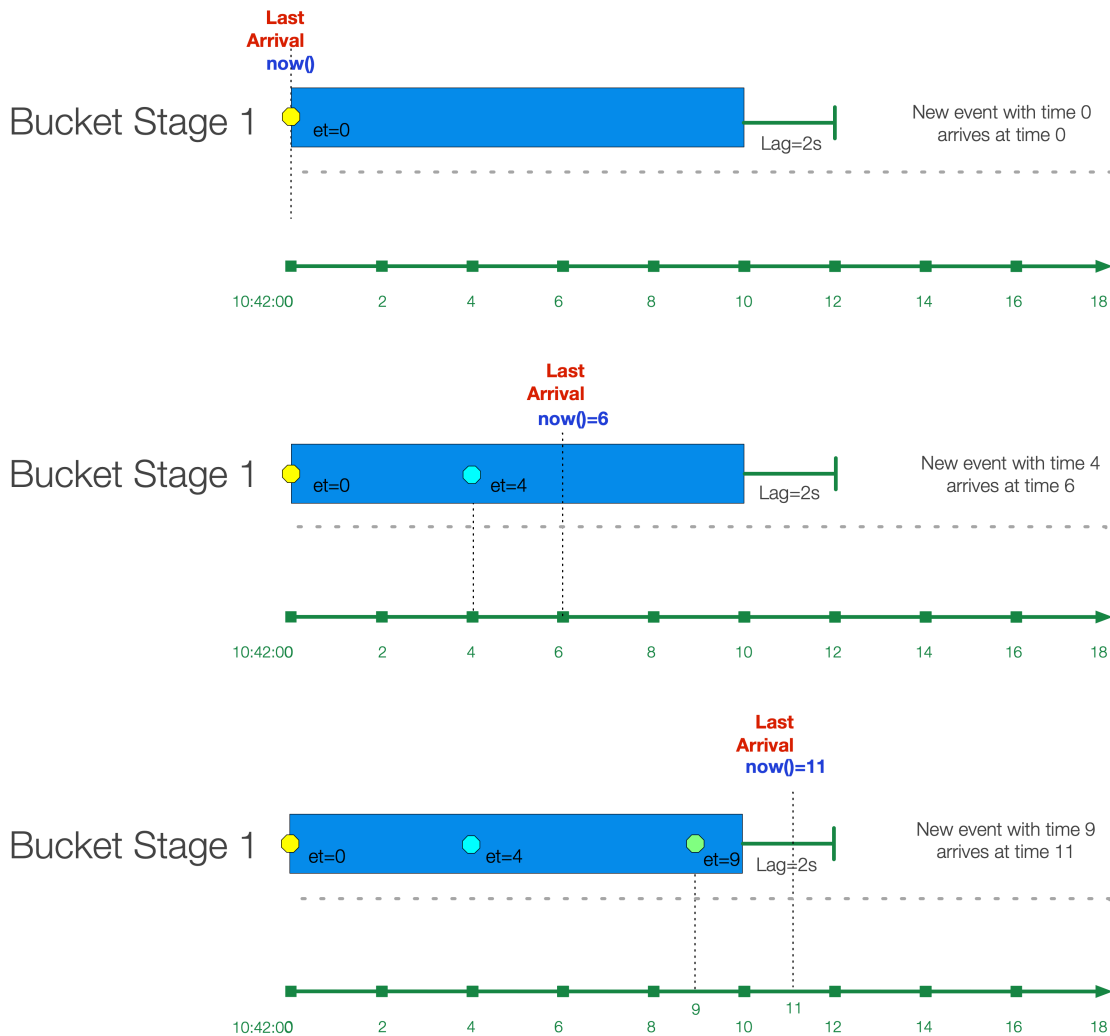
`variancep(expr:FieldExpression)` : Returns the population variance of the values of the parameter.

How Do Time Window Settings Work?

Lag Tolerance

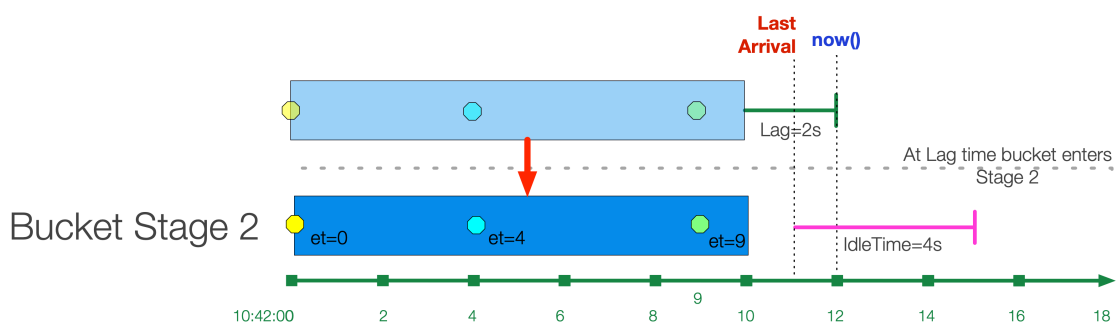
As events are aggregated into windows, there is a good chance that most will arrive later than their event time. For instance, given a 10s window (10:42:00 – 10:42:10), an event with timestamp 10:42:03 might come in 2 seconds later at 10:42:05 .

In several cases, there will also be late, or lagging, events that will arrive **after** the latest time window boundary. For example, an event with timestamp 10:42:04 might arrive at 10:42:12 . Lag Tolerance is the setting that governs how long to wait – after the latest window boundary – and still accept late events.



The "bucket" of events is said to be in Stage 1, where it's still accepting new events, but it's not yet finalized. Notice how in the third case, an event with event time 10:42:09 arrives 1 second past the window boundary at 10:42:11, but it's still accepted because it happens before the lag time expires.

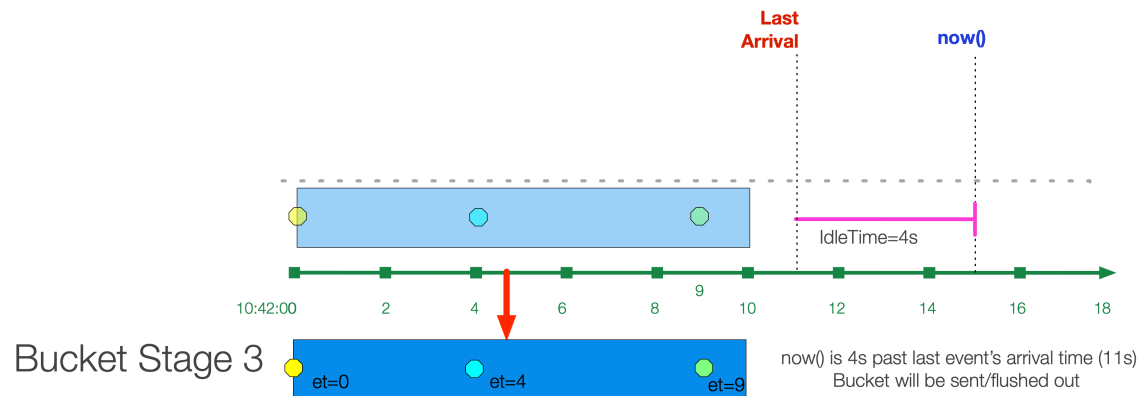
After Lag time expires, the bucket moves to Stage 2.



If the bucket is created from a historic stream, then the bucket is initiated in Stage 2. Lag time is not considered. A "historic" stream is one where the latest time of a bucket is before `now()`. E.g., if the window size is 10s, and `now()`=10:42:42, an event with `event_time=10` will be placed in a Stage 2 bucket with range 10:42:10 – 10:42:20.

Idle Bucket Time Limit

While Lag Tolerance works with event time, Idle Bucket Time Limit works on arrival time (i.e., real time). It is defined as the amount of time to wait before flushing a bucket that has not received events.



After the Idle Time limit is reached, the bucket is "flushed" and sent out of the system.

Examples

Assume we're working with VPC Flowlog events that have the following structure:

```
version account_id interface_id srcaddr dstaddr srcport dstport protocol
packets bytes start end action log_status
```

For example:

```
2 99999XXXXX eni-02f03c2880e4aaa3 10.0.1.70 10.0.1.11 9999 63030 6 6556 262256
1554562460 1554562475 ACCEPT OK
2 496698360409 eni-08e66c4525538d10b 37.23.15.38 10.0.2.232 4373 8108 6 1 52
1554562456 1554562466 REJECT OK
```

Scenario A: Every 10s, compute sum of `bytes` and output it in a field called `TotalBytes`.

Time Window: 10s

Aggregations: `sum(bytes).as(TotalBytes)`

Scenario B: Every 10s, compute sum of `bytes`, output it in a field called `TotalBytes`, group by `srcaddr`.

Time Window: 10s

Aggregations: `sum(bytes).as(TotalBytes)`

Group by Fields: `srcaddr`

Scenario C: Every 10s, compute sum of `bytes` but only where action is `REJECT`, output it in a field called `TotalBytes`, group by `srcaddr`.

Time Window: 10s

Aggregations: `sum(bytes).where(action=='REJECT').as(TotalBytes)`

Group by Fields: `srcaddr`

Scenario D: Every 10s, compute sum of `bytes` but only where action is `REJECT`, output it in a field called `TotalBytes`. Also, compute distinct count of `srcaddr`.

Time Window: 10s

Aggregations:

`sum(bytes).where(action=='REJECT').as(TotalBytes)`

`distinct_count(srcaddr).where(action=='REJECT')`

 Updated 11 days ago

CEF Serializer

Description

The `CEF Serializer` takes a list of fields and/or values, and formats them in Common Event Format (CEF) standard.

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 be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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. Fields 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 expression, or can be constants.

 Updated 11 days ago

Clone

Description

The `Clone` function clones events, with optional added fields.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.


Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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

 Coming soon.

os

 Updated 11 days ago

Comment

Description

The `Comment` function adds a text comment in the pipeline

Usage

Comment: Text input field to add comment.

 Updated 7 months ago

Drop

Description

The `Drop` function will drop/delete any events that meet the Filter expression.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

 Updated 11 days ago

Dynamic Sampling

Description

The `Dynamic Sampling` function filters out events based on an expression, a sample mode, and volume.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Sample Mode: Defines how sample rate will be derived. Supported methods:

- Square Root: `sqrt(previousPeriodCount)`.
- Logarithmic: `log(previousPeriodCount)`. Defaults to Logarithmic.

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 volume. 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 clamped down to it.

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 incoming data volume per sample group. This function allows users to set only the aggressiveness/coarseness of this adjustment. Square Root is more aggressive than Logarithmic setting.

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 `Math.ceil(Math.log(lastPeriodVolume))` (natural log). 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.

 Updated 11 days ago

Eval

Description

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 be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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 (can be a constant). Nested addressing is supported.

Keep Fields: List of fields to keep. Wildcards (*) and nested addressing are supported. Takes precedence over **Remove Fields** (below).

Remove Fields: List of fields to remove. Wildcards (*) and nested addressing are supported. Cribl LogStream 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

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 a multivalued field `myTags` :

- **Name:** `myTags`
- **Value Expression:** `login=='error' ? [...myTags, 'error'] : myTags`

 See [Ingest-time Fields](#) for more examples.

Advanced Usage Notes

Note 1

The Eval function has the ability to 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.

- Simple Example: `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.
- Another Example: 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.

Note 2

You can also use the Eval function to set and unset control fields (e.g., `_TCP_ROUTING` in Splunk) via this syntax: `_ctrl.<name>` . Control fields can be referenced only on the left-hand side of **Add**. (I.e., they cannot be read or used on the right-hand side, and cannot be referenced in **Remove**.)

To unset/delete a control field, set its value to `undefined` . These fields are normally not needed for event computations, and modifying them **is suggested to be done only by experts**. Please reach out to Cribl team if you need help with this topic.



Updated 11 days ago

Flatten

Description

The `Flatten` function is used to flatten fields out of a nested structure.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Fields: List of top-level fields to include for flattening. Supports `*` wildcards. Defaults to empty array, which means all fields.

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 be used for flattening. Defaults to `_` (underscore).

Example

Assume an input event like this:

```
input

{ "accounting" : [ { "firstName" : "John", "lastName" : "Doe", "age" : 23 }, {
```

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,  
}
```

 Updated 10 days ago

GeoIP

Description

The `GeoIP` function enriches events with geo fields, given an IP address. It is optimized for binary databases such as [Maxmind's GeoIP](#).

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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, the **GeoIP File** does not need to be an absolute path.

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`.

 Updated 10 days ago

JSON Unroll

Description

The `JSON Unroll` function accepts a `_raw` field as a JSON string, and unrolls/explodes an **array of objects** from the field into individual events.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Path: Path to array to unroll, e.g., `foo.0.bar`.

New Name: The name of each element in the new event. Leave empty to expand the array element.

Examples

Assume you have an incoming event that has a `_raw` field as a JSON 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
```

Event 2:

```
{"_raw":{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name
```

Event 3:

```
{"_raw":{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name
```

Event 4:

```
{"_raw":{"date":"9/25/18 9:10:13.000 PM","name":"Amrit","age":42,"cars":{"name
```

 Updated 10 days ago

Lookup

Description

The `Lookup` function enriches events with external fields. CSV lookup table files are supported.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Lookup file path (.csv, .csv.gz): Path to the location of the lookup file. Environment variables can be referenced via `$`, e.g.: `$HOME/file.csv`.

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, this attribute further 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 output as arrays. Defaults to `First Match`.

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 event field name. This input is optional.

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

Add to raw event: Whether to append the looked-up values to the `_raw` field as, key=value pairs. Defaults to No .

Example 1: Regex Lookups

paloalto.csv

Match Mode: Regex

Match Type: First Match

Lookup Field Name in Event: `_raw`

Corresponding Field Name in Lookup: regex

Events before and after

BEFORE:

```
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,F00BAR,TRAFFIC,end,2049,2018-09-20T13:03:58.000Z"}
{"_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,F00BAR,THREAT,end,2049,2018-09-20T13:03:58.000Z"}
```

AFTER:

```
{ "_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,F00BAR,TRAFFIC,end,2049,2018-09-20T13:03:58.000Z",
  "sourcetype": "pan:traffic"
}

{ "_raw": "Sep 20 13:03:55 PA-VM 1,2018/09/20 13:03:58,F00BAR,THREAT,end,2049,2018-09-20T13:03:58.000Z",
  "sourcetype": "pan:threat"
}
```

Example 2:

CIDR Lookups - Assign a `location` field to events if their `destination_ip` field matches a particular CIDR range.

paloaltoips.csv


```
range, location
10.0.0.0/24, San Francisco
10.0.0.0/16, California
10.0.0.0/8, US
```

Match Mode: CIDR

Match Type: See options below

Lookup Field Name in Event: destination_ip

Corresponding Field Name in Lookup: range

 In Match Mode CIDR with Match Type **Most Specific**, the lookup will implicitly search for matches from most specific to least specific. There is no need to pre-sort data.

Note that Match Mode CIDR with Match Type **First Match** is likely the most performant with large lookups. This can be used as an alternative to Most Specific, if the file is sorted with the most specific/relevant entries first. This mode still performs a table scan, top to bottom.

Events before and after

BEFORE:

```
{"_raw": "Sep 20 13:03:55 PA-VM 1, 2018/09/20 13:03:58, F00BAR, TRAFFIC, end, 2049",
  "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, F00BAR, TRAFFIC, end, 2049",
  "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, F00BAR, TRAFFIC, end, 2049",
  "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, F00BAR, TRAFFIC, end, 2049",
  "destination_ip": "10.0.0.102",
  "location": [
    "San Francisco",
    "California",
  ]
}
```

```
"US",  
  ]}
```

See [Ingest-time Lookups](#) for other examples.

 Updated 10 days ago

Mask

Description

The `Mask` function masks, or replaces patterns in events.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Masking Rules:

Match Regex and Replace Expression pairs. Defaults to empty.

- **Match Regex:** Pattern to replace. Use `/g` to replace all matches, e.g.: `/(bar)/g`
- **Replace Expression:** A JS expression or literal to replace the matching content.

Apply to Fields: Fields on which to apply the masking rules. Defaults to `_raw`. Wildcards (*) and nested addressing are supported.



Negated terms are also supported. The list is order-sensitive when negated terms are used. E.g., `!foobar`, `foo*` means "All fields that start with 'foo', except 'foobar'." `!foo*`, `*` means "All fields, except for those that start with 'foo!'."

Examples

See [Masking and Obfuscation](#) for examples.

 Updated 10 days ago

Numerify

Description

The `Numerify` function converts event fields that are numbers to type `number` .


Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No` .

Ignore Fields: Specify fields to **not** numerify, one field per row. By default, numerify will apply to **all** fields. Wildcards (*) and nested addressing are supported.

 Negated terms are also supported. The list is order-sensitive when negated terms are used. E.g., `!foobar`, `foo*` means "All fields that start with 'foo', except 'foobar'." `!foo*`, `*` means "All fields, except for those that start with 'foo'."

 Updated 10 days ago

Parser

Description

The `Parser` function can be used to extract fields out of events, or to reserialize (re-write) events with a subset of fields. Reserialization will **maintain** the format of the event. For example, if an event contains comma-delimited fields, and `fieldA` and `fieldB` are filtered out, those fields' positions will be set to null and not deleted completely.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Parser Mode: Operating mode. **Extract** creates new fields. **Reserialize** will extract, filter fields, and then reserialize. **Serialize** will put fields in a certain format. Defaults to `Extract`.

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. `Extract` and `Serialize Mode` only.


Type: Parser/Formatter type to use. Options: `CSV`, `JSON`, `K=V Pairs`, `Extended Log File Format (ELFF)`, `Common Log Format (CLF)`.

Library: Browse Parser/Formatter library.

List of Fields: Fields expected to be extracted, in order. If not specified, parser will auto-generate.

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**. The list is order-sensitive when negated terms are used. E.g., `!foobar`, `foo*` means

"All fields that start with 'foo', except 'foobar'." !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.

Destination Field: Field in which to add extracted fields (Extract mode only).

How Do Fields to Keep, Fields to Remove, and Fields Filter Expression Interact

Order of priority: **Fields to Keep** > **Fields to Remove** > **Fields Filter Expression**

If a field is in **Fields to Keep** and **Fields to Remove**, **Fields to Keep** takes precedence.

If a field is in **Fields to Remove** and in **Fields Filter Expression**, **Fields to Remove** takes precedence.

Example 1

Assume we have an event with KV pairs as below:

```
<timestamp> a=000,b=001,c=002,d=003,e=004,f=005,g1=006,g2=007,g3=008, ...
```

To extract all fields, we can set the Parser Type to K=V 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:

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:
- 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:
- 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:
- 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 those that end in `Cxn` in the `values.total.` path.

Parser Function Configuration:

Assume we have an event with KV pairs as below:

```
<timestamp> 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 K=V Pairs. Then proceed with another Parser function right below it.

Scenario A: Serialize fields `a`, `b`, `c`, `d` in CSV format

Expected result: `_raw` field will have this value `000,001,002,003`

Parser 2

- Operation Mode: Serialize
- Source Field:
- Destination Field:
- Type: CSV
- List of Fields: `a`, `b`, `c`, `d` (needed for positional formats)

Scenario B: Serialize fields `a`, `b`, `c` in JSON format, under a field called `bar`

Expected result: `bar` field will be set to: `{"a":"000","b":"001","c":"002","d":"003"}`

Parser 2

- Operation Mode: Serialize
- Source Field:
- Destination Field: `bar`
- Type: JSON
- List of Fields:
- Fields to Keep: `a`, `b`, `c`, `d`

 Updated 9 days ago

Publish Metrics

Description

The `Publish Metrics` function extracts, formats and outputs metrics from events.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty - all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Metrics: List of metrics from event to extract and format. Formatted metrics can be used by a destination to pass metrics to a metrics aggregation platform.

- **Event Field Name:** The name of the field in event containing the metric value.
- **Metric Name Expression:** JavaScript expression to evaluate metric field name. Defaults to Event Field Name.
- **Metric Type:** Type of metric.

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. Defaults to `!_* *`.



Dimensions supports wildcards and negated terms. The list is order-sensitive when negated terms are used. E.g., `!foobar, foo*` means "All fields that start with 'foo', except 'foobar'." `!foo*, *` means "All fields, except for those that start with 'foo'."

Overwrite: If true, overwrite previous metric specs; otherwise, append. Defaults to `No`.

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
```

... and we want to use values of `bytes` and `packets` as metrics across these dimensions:
`action`, `interface_id`, and `dstaddr`.

Metrics:

Event Field Name	Metric Name Expression	Metric Type
bytes	<code>`metric_name.bytes`</code>	Gauge
packets	<code>`metric_name.packets`</code>	Gauge

Dimensions:

Dimensions
<code>action interface_id dstaddr</code>

OUTPUT

```
{
  "action": "REJECT",
  "interface_id": "eni-02f03c2880e4aaa3",
  "dstaddr": "10.0.1.11",
  "metric_name.bytes": 262256,
  "metric_name.packets": 6556,
}
```

 Updated 9 days ago

Regex Extract

Description

The `Regex Extract` function extracts fields with regex named groups. (In Splunk, these will be index-time fields). Fields that start with `__` (double underscore) are special fields in Cribl LogStream. 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 be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description about this function. Defaults to empty.

Final: If true, stops data from being fed 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 name and value** of a field, e.g.: `(?<__NAME_0>[^\s=]+)=(?<__VALUE_0>[^\s]+)`. Defaults to empty. See [Examples](#) below.

Source Field: Field in 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. Used by `__NAME_N` and `__VALUE_N` capturing groups. Named capturing groups will always use a value of `1`. Defaults to `100`.

Field Name Format Expression: Expression to format field names when `__NAME_` capturing groups are used. The **original** field name is in the global `name`. E.g., to append `xx` to all field names: ``_${name}_XX``. If not specified, names will be sanitized using regex: `/^[_0-9]+|^[a-zA-Z0-9_]+/g`.

Overwrite Existing Fields: 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 and 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

Assume a simple event that looks like this: `metric1=23 metric2=42 dc=23 abc=xyz`

1. Extract only the `metric1` field:

Regex: `metric1=(?<metric1>\d+)`

Result: `metric1:"23"`

2. Extract all `k=v` pairs:

Regex: `(?<_NAME_0>[^\s]+)=(?<_VALUE_0>[^\s]+)`

Result: `metric1:"23" , metric2:"42" , dc"23" , abc:"xyz"`



Updated 9 days ago

Regex Filter

Description

The `Regex Filter` function will filter out events based on regex matches.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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.

 Updated 5 days ago

Sampling

Description

The `Sampling` function filters out events, based on an expression and a sampling rate.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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.

Examples

See [Sampling](#) for examples.

 Updated 5 days ago

Serialize

Description

Use the `Serialize` function to serialize the content of an event into a predefined format.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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. (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

Assume a simple event that looks like this: `{"time":"2019-08-25T14:19:10.240Z","channel":"input","level":"info","message":"initializing input","type":"kafka"}`

Serialize these fields: `_time`, `channel`, `level`, `type`, in CSV format, into a new destination field called `test`.

Type: `CSV`

Fields to Serialize: `_time` `channel` `level` `type`

Destination Field: `test`

Result: `_raw: 1566742750.24,input,info,kafka`



 Updated 5 days ago

Suppress

Description

The `Suppress` function suppresses events over a period of time based on a key expression evaluation.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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`.

Advanced Settings

Maximum Cache Size : The maximum number of keys that can be cached before idle entries are removed. Leave at the default `50000` unless you understand the implications of changing this.

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. Leave at the default `2` unless you understand the implications of changing.

Num Events to Trigger Cache Clean-Up: Threshold number of events at which to check cache for idle sessions, when cache exceeds the 'Maximum Cache Size'. Leave at the default `10000` unless you understand the implications of changing.

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): `300`

Result: One event per unique `host` value will be allowed in every 300s. 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`.

- To **guarantee** that suppression applies **only** to events with `host` and `port`, check for their presence using Filter:

Filter: `host!=undefined && port!=undefined`

Key Expression: ``${host}:${port}``

Number to Allow: `1`

Suppression Period (sec): `300`

3. 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.

 Updated 5 days ago

Tee

Description

The `Tee` function tees events out to a command of choice, via `stdin` – one JSON-formatted event per line.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed to the downstream functions. Defaults to `No`.

Command: Command to execute and feed events to (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



Coming soon.

 Updated about 16 hours ago

XML Unroll

Description

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 be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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: LogStream will add a field with this name, containing the index (starting from 0) at which the item was located. 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 you have an incoming event as below, and want to break all the `Child` elements and inherit `myID` and `branchLocation`,

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>

```

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>

```

 Updated 5 days ago

Prometheus Publisher (beta)

Description

The `Prometheus Publisher` function allows for metrics to be published to a Prometheus-compatible metrics endpoint.

In the current implementation, the endpoint is: `<cribl-host>:<api-port>/metrics` .



The function should **follow** the [Publish Metrics](#) or [Aggregations](#) functions.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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: Determines whether the event should be consumed once published. Defaults to `No` .

Update Mode: Determines whether the publisher updates (versus overwrites) the published output. Defaults to `Yes` . Toggle to `No` to overwrite.

 Updated 5 days ago

Reverse DNS (beta)

Description

The `Reverse DNS` function resolve hostnames, using an IP address.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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 (in minutes) to refresh the DNS cache. Use `0` to disable. Defaults to `60`.

 Updated 5 days ago

Unroll

Description

The `Unroll` function accepts an array field – or an expression to evaluate an array field – to break/unroll into individual events.

Usage

Filter: Filter expression (JS) that selects data to be fed through the function. Defaults to empty, meaning that all events will be evaluated.

Description: Simple description of this function. Defaults to empty.

Final: If true, stops data from being fed 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(s)

Assume we want to break/unroll each line of this event:

Sample Event

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
root	1	0.0	0.5	38000	5356	?	Ss	2018	2:02	/lib/systemd/
root	2	0.0	0.0	0	0	?	S	2018	0:00	[kthreadd]
root	3	0.0	0.0	0	0	?	S	2018	1:51	[ksoftirqd/0]
root	5	0.0	0.0	0	0	?	S<	2018	0:00	[kworker/0:0H]
root	7	0.0	0.0	0	0	?	S	2018	3:55	[rcu_sched]
root	8	0.0	0.0	0	0	?	S	2018	0:00	[rcu_bh]

Settings

Source Field Expression: `_raw.split(/\n/)`



The `split()` JavaScript method breaks `_raw` into an ordered set of substrings/values, puts these values into an array, and returns the array.



Destination Field: `_raw`

Resulting Events

Event 1:

USER	PID	%CPU	%MEM	VSZ	RSS	TTY	STAT	START	TIME	COMMAND
------	-----	------	------	-----	-----	-----	------	-------	------	---------

Event 2:

root	1	0.0	0.5	38000	5356	?	Ss	2018	2:02	/lib/systemd/s
------	---	-----	-----	-------	------	---	----	------	------	----------------

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:

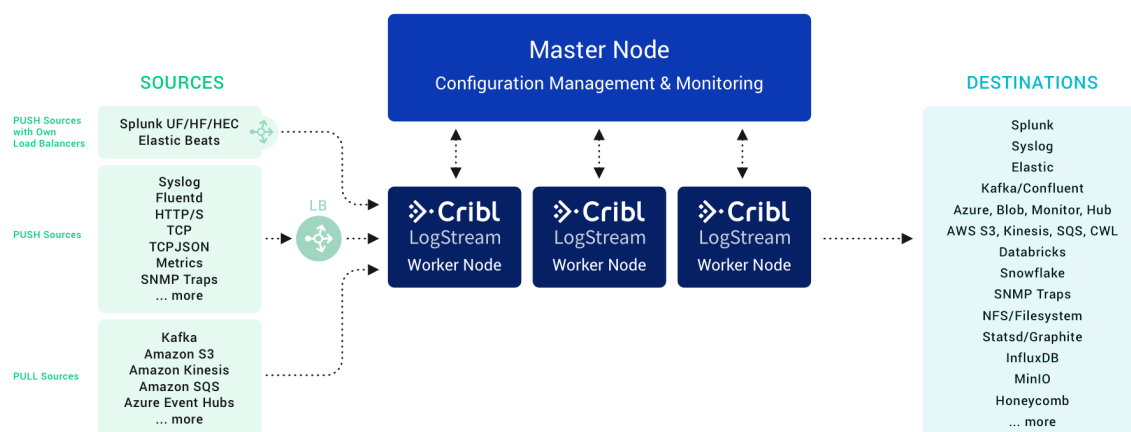
root	8	0.0	0.0	0	0	?	S	2018	0:00	[rcu_bh]
------	---	-----	-----	---	---	---	---	------	------	----------



Updated 5 days ago

Sources

Cribl LogStream can receive data from various sources, including Splunk, HTTP, Elastic Beats, Kinesis, Kafka, TCP JSON, and many others.



PUSH Sources

Supported data sources that **send** to Cribl LogStream:

- Splunk TCP
- Splunk HEC
- Elasticsearch
- Syslog
- TCP JSON
- TCP JSON
- HTTP(S)
- SNMP Traps
- Metrics

Data from these sources is normally sent to a set of LogStream Workers through a loadbalancer. Some sources, such as [Splunk](#) forwarders, have native loadbalancing capabilities, and these should therefore be pointed directly at LogStream.

PULL Sources

Supported sources that Cribl LogStream **fetches** data from:

- Kafka
- Kinesis Streams

- Azure Event Hubs
- SQS
- S3

Internal Sources

Sources that are **internal** to Cribl LogStream:

- Datagens
- Cribl Internal

Configuring and Managing Sources

For each source *type*, you can create multiple definitions, depending on your requirements. To configure sources, click on **Sources**, select the desired type from the left menu, then click **Add New**.

 Updated 4 days ago

Splunk TCP

Cribl LogStream supports receiving Splunk [data](#) from Universal or Heavy Forwarders.

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Splunk Data

While on the **Sources** screen, select **Splunk** > **Splunk TCP** from the left menu, then click **Add New**. The resulting **New Splunk source** pane provides the following fields.

Source 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.
- **IP Whitelist Regex:** Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e., all IPs).

TLS Settings (Server Side)

- **Enabled** defaults to `No`. When toggled to `Yes`:
- **Certificate Name**: Name of the predefined certificate.
- **Private Key Path**: Path on server where to find the private key to use in PEM format. Path can reference `$ENV_VARS`.
- **Passphrase**: Passphrase to use to decrypt private key.
- **Certificate Path**: Path on server where to find certificates to use, in PEM format. Path can reference `$ENV_VARS`.
- **CA Certificate Path**: Path on server where to find CA certificates to use in PEM format. 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`:
 - **Common Name**: Regex matching peer certificate subject common names allowed to connect. Defaults to `.*`.
- **Validate Client Certs**: Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No`.

Advanced Settings

- **Enable Proxy Protocol:** Defaults to `No` . Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.

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:** The amount of time (in milliseconds) that 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. Defaults to `10000` .

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

`__inputId`

Configuring A Splunk Forwarder

To configure a Splunk forwarder (UF, HF) use the following [outputs.conf](#) stanzas.

```
.../outputs.conf
```

```
[tcpout]
disabled = false
defaultGroup = cribl, <optional_clone_target_group>,
```

```
[tcpout:cribl]
server = [<cribl_ip>|<cribl_host>]:<port>, [<cribl_ip>|<cribl_host>]:<port>, ..
sendCookedData=true
useACK = false
negotiateNewProtocol = false
negotiateProtocolLevel = 0
```

 Updated 26 days ago

Splunk HEC

Cribl LogStream supports receiving data over HTTP/S using [Splunk HEC](#).

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Data over Splunk HEC

While on the **Sources** screen, select **Splunk > HEC** from the left menu, then click **Add New**. The resulting **New Splunk HEC source** pane provides the following fields.

Source 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.
- **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. Defaults to `/services/collector`.
- **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`.

Auth Tokens

- **Token:** Shared secrets to be provided by any client (Authorization: `<token>`). If empty, unauthenticated access **will be permitted**.
- **Fields:** Fields (metadata) to add to events referencing this token. Each field is a **Name/Value** pair.

These fields may be overridden by fields added at the request level.

TLS Settings (Server Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Certificate Name:** The 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` :
 - **Common Name:** Regex matching peer certificate subject common names allowed to connect. Defaults to `.*` .
- **Validate Client Certs:** Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No` .

Advanced Settings

- **Enable Proxy Protocol:** Defaults to `No` . Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.

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:** The amount of time (in milliseconds) that 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. Defaults to `10000` .

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

These fields may be overridden by fields added at the token or request level.

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__inputId
__hecToken

Format and Endpoint Example

- Configure Cribl LogStream to listen on port `10080` with an authToken of `myToken42` .
- Send a payload to your Cribl LogStream receiver.

Note: Token specification can be either `Splunk <token>` or `<token>`

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
```

```
curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myTok
```

 Updated 26 days ago

Syslog

Cribl LogStream supports receiving of data over syslog.

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Data over Syslog

While on the **Sources** screen, select **Syslog** from the left menu, then click **Add New**. The resulting **New Syslog source** pane provides the following fields.

Source 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.
- **TCP Port:** Enter the TCP port number to listen on. Not required if listening on UDP.

TLS Settings (TCP Only)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Certificate Name:** The 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`:
 - **Common Name:** Regex matching peer certificate subject common names allowed to connect. Defaults to `.*`.
- **Validate Client Certs:** Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No`.

Advanced Settings

- **Enable Proxy Protocol:** Defaults to `No` . Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.
- **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.
- **Default Timezone:** Timezone to assign to timestamps without timezone info. Defaults to `local` .
- **Single Msg Per UDP:** Whether to treat UDP packet data received as full syslog message. Defaults to `No` . (i.e., newlines in the packet will be treated as event delimiters.)

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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.

Field(s) for this source:

`__inputId`
`__srcIpPort`

 Updated 26 days ago

Elasticsearch

Cribl LogStream supports receiving data over HTTP/S using the [Elastic Bulk API](#). (See the [Configuring Filebeat](#) example below.)

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Data over HTTP(S) Using the Elastic Bulk API Protocol

While on the **Sources** screen, select **Elasticsearch** from the left menu, then click **Add New**. The resulting **New Elastic source** pane provides the following fields.

Source 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.
- **Auth Tokens:** Shared secrets to be provided by any client (Authorization: <token>). If empty, unauthenticated access **will be permitted**.
- **Elastic API Endpoint (Bulk API):** Absolute path on which to listen for Elastic API requests. Currently, only `_bulk` (default: `/elastic/_bulk`) is available; others are faked as success. Use an empty string to disable. Default entry is `/elastic`.

TLS Settings (Server Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Certificate Name:** The 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`:
 - **Common Name:** Regex matching peer certificate subject common names allowed to connect. Defaults to `.*`.
- **Validate Client Certs:** Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No`.

Advanced Settings

- **Enable Proxy Protocol:** Defaults to `No`. Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Field Normalization

The Elasticsearch 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 LogStream 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(s) for this source:

`__inputId`

```
__id
__type
__index
__host
```

Configuring Filebeat

To set up Filebeat to send data to LogStream, use its [Elasticsearch output](#). If an Auth Token is configured here, it should be added on Filebeat configuration under `output.elasticsearch.headers` .

```
...filebeat.yml
```

```
output.elasticsearch:
  # Array of hosts to connect to.
  hosts: ["http://<LOGSTREAM_HOST>:9200/elastic"]
```

```
output.elasticsearch.headers:
  Authorization: "myToken42"
```

 Updated 26 days ago

TCP JSON

Cribl LogStream supports receiving of data over TCP in JSON format (see protocol below).

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive TCP JSON Data

While on the **Sources** screen, select **TCP JSON** from the left menu, then click **Add New**. The resulting **New TCP JSON source** pane provides the following fields.

- **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 port number.
- **IP Whitelist Regex:** Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e., all IPs).
- **Shared Secret (authToken):** Shared secret to be provided by any client (in `authToken` header field). If empty, unauthenticated access will be permitted.

Source Settings

TLS Settings (Server Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Certificate Name:** The 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`:
 - **Common Name:** Regex matching peer certificate subject common names allowed to connect. Defaults to `.*`.

- **Validate Client Certs:** Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No`.

Advanced Settings

- **Conditioning Pipeline:** Pipeline to process data from this input before being sent through the routes.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

`__inputId`

Format

LogStream 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 it 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"}  
{"host":"myOtherHost", "source":"myOtherSource", "_raw": "{\"message\":\"Something\"}"}
```

TCP JSON Field Mapping to Splunk

if a TCP JSON source is routed to a Splunk destination, fields within the JSON payload are mapped to Splunk fields. Fields that do not have corresponding (native) Splunk fields become index-time fields. For example, let's assume we have a TCP JSON event as below:

```
{"_time":1541280341, "host":"myHost", "source":"mySource", "_raw":"this is a  
sample event ", "fieldA":"valueA"}
```

Here, `_time`, `host` and `source` become their corresponding fields in Splunk. The value of `_raw` becomes the actual body of the event, and `fieldA` becomes an index-time field.
(`fieldA::`valueA``)

Example

1. Configure Cribl LogStream 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 LogStream host: `cat test.json | nc <myCriblHost> 10001`

 Updated 26 days ago

TCP (RAW)

Cribl LogStream supports receiving of data over TCP (see examples and header options below).

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive TCP Data

While on the **Sources** screen, select **TCP** from the left menu, then click **Add New**. The resulting **New TCP source** pane provides the following fields.

Source Settings

- **Input ID:** Enter a unique name to identify this TCP source definition.
- **Address:** Enter hostname/IP to listen for TCP JSON data. E.g. `localhost` or `0.0.0.0`.
- **Port:** Enter port number.
- **IP Whitelist Regex:** Regex matching IP addresses that are allowed to establish a connection. Defaults to `.*` (i.e., all IPs).
- **Enable Header:** Enable to indicate header record will be passed by client with every new connection. The header can contain `authToken` and 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" : "myToken42", "fields": { "field1": "value1", "field2": "value2" } }<NewLine>
```

 Defaults to `No`.
 - **Shared secret (authToken):** Shared secret to be provided by any client (in `authToken` header field). If empty, unauthenticated access will be permitted.

TLS Settings (Server Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Certificate Name:** The 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` :
 - **Common Name**: Regex matching peer certificate subject common names allowed to connect. Defaults to `.*` .
- **Validate Client Certs**: Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No` .

Advanced Settings

- **Enable Proxy Protocol**: Defaults to `No` . Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.

Processing Settings

Custom Command Processor

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**: The amount of time (in milliseconds) that 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. Defaults to `10000` .

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) accessible for this source:

```
__inputId
__srcIpPort
__channel
```

TCP Source Example

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.

Sample raw TCP test

```
{"authToken":"myToken42", "fields": {"region": "us-east-1", "AZ":"az1"}}
this is event number 1
this is event number 2
```

Enabling the Example

1. Configure LogStream 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 LogStream host, using this command: `cat test.raw | nc <myCriblHost> 7777`

 Updated 26 days ago

HTTP(S)

Cribl LogStream supports receiving data over HTTP/S using the Cribl Bulk API, [Splunk HEC](#), or [Elastic Bulk API](#).

 Type: **Push** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Data over HTTP(S)

While on the **Sources** screen, select **HTTP** from the vertical menu, then click **Add New**. The resulting **New HTTP source** pane provides the following fields.

Source 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.
- **Auth Tokens:** Shared secrets to be provided by any client (Authorization: <token>). If empty, unauthenticated access **will be permitted**.
- **Cribl HTTP Event API:** Absolute path on which to listen for Cribl HTTP API requests. Currently, only `_bulk` (default `/cribl/_bulk`) is available. Use an empty string to disable. Default entry is `/cribl`. Max payload size is 2MB.
- **Elastic API Endpoint (Bulk API):** Absolute path on which to listen for Elastic API requests. Currently, only `_bulk` (default: `/elastic/_bulk`) is available; others are faked as success. Use an empty string to disable. Default entry is `/elastic`.

Notes

This Elastic API implementation is **superseded** by the Cribl [Elasticsearch](#) Source.

- **Splunk HTTP Event Collector API:** 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`.
- **Splunk HEC Acks:** Whether to enable Splunk HEC acknowledgements. Defaults to `No`.

Notes

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.)

This implementation is **superseded** by the Cribl [Splunk HEC](#) Source.

TLS Settings (Server Side)

- **Enabled** defaults to `No` . When toggled to `Yes` :
- **Certificate Name**: The 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` :
 - **Common Name**: Regex matching peer certificate subject common names allowed to connect. Defaults to `.*` .
- **Validate Client Certs**: Require server to reject any connection that is not authorized with the list of supplied CAs. Defaults to `No` .

Advanced Settings

- **Enable Proxy Protocol**: Defaults to `No` . Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name**: Field name.
- **Value**: JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

```
__inputId
__id (Elastic In)
__type (Elastic In)
__index (Elastic In)
__host (Elastic In)
```

Format and Endpoint

LogStream 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":
{"_time":1541280341, "host":"myOtherHost", "source":"myOtherSource", "_raw": ".
```

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

For the following examples:

1. Configure Cribl to listen on port `10080` for HTTP (default). Set `authToken` to `myToken42`.
2. Send a payload to your Cribl LogStream receiver.

Cribl Endpoint – Single Event

Cribl Single Event Example:

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d
```

Cribl Endpoint – Multiple Events

Cribl Endpoint - Multiple Events

```
curl -k http://<myCriblHost>:10080/cribl/_bulk -H 'Authorization: myToken42' -d
```

Splunk HEC Event Endpoint

Splunk HEC Event Endpoint

```
curl -k http://<myCriblHost>:10080/services/collector/event -H 'Authorization:
```

```
curl -k http://<myCriblHost>:10080/services/collector -H 'Authorization: myTok
```



For Splunk HEC, the token specification can be either Splunk <token> or <token> .



Updated 18 days ago

Kafka

Cribl LogStream supports receiving data records from a [Kafka](#) cluster.

 Type: **Pull** | TLS Support: **YES**

Configuring Cribl LogStream to Receive Data from Kafka Topics

While on the **Sources** screen, select **Kafka** from the left menu, then click **Add New**. The resulting **New Kafka source** pane provides the following fields.

Source Settings

- **Input ID:** Enter a unique name to identify this source definition.
- **Brokers:** List of Kafka brokers to use, e.g., `localhost:9092` .
- **Topics:** List of topics to subscribe to.
- **Group ID:** The name of the consumer group to which this Cribl LogStream instance belongs.
- **From beginning:** Whether to start reading from the earliest available data. Relevant only during initial subscription. Defaults to `Yes` .

TLS Settings (Client Side)

- **Enabled:** defaults to `No` . When toggled to `Yes` :
- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

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 :
- **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:



These have the same format as the [TLS Settings \(Client Side\)](#) above.

TLS Settings (Schema Registry)

- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

Authentication

This section governs SASL (Simple Authentication and Security Layer) authentication.

- **Enabled:** Defaults to No . When toggled to Yes :
- **SASL Mechanism:** Use this drop-down to select the SASL authentication mechanism to use.
- **Username:** Enter the username for your account.
- **Password:** Enter the account's password.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

`__inputId`

`__topicIn` (indicates the Kafka topic that the event came from.; see `__topicOut` in our [Kafka Destination](#) documentation)

`__schemaId` (when using Schema Registry)

 Updated 25 days ago

Kinesis Streams

Cribl LogStream supports receiving data records from [Amazon Kinesis Streams](#).

 Type: **Pull** | TLS Support: **YES** (secure API)

Configuring Cribl LogStream to Receive Data from Kinesis Streams

While on the **Sources** screen, select **Kinesis** from the left menu, then click **Add New**. The resulting **New Kinesis source** pane provides the following fields.

Source 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.
- **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`.
- **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 , CloudWatch Logs, Event per line, and New Line JSON). Defaults to `Cribl`.
- **API Key:** If not present, will fall back to `env.AWS_ACCESS_KEY_ID` , or to the metadata endpoint for IAM credentials.
- **Secret Key:** If not present, will fall back to `env.AWS_SECRET_ACCESS_KEY` , or to the metadata endpoint for IAM credentials.
- **Region:** Region where the Kinesis stream is located. Required.

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`.

- **Verify KPL Checksums:** Enable this setting to verify Kinesis Producer Library (KPL) event checksums.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__inputId

 Updated 26 days ago

Azure Event Hubs

Cribl LogStream supports receiving data records from [Azure Event Hubs](#).

 Type: **Pull** | TLS Support: **YES** (secure API)

Configuring Cribl LogStream to receive data from Azure Event Hubs.

While on the **Sources** screen, select **Azure Event Hubs** from the left menu, then click **Add New**. The resulting **New Event Hubs source** pane provides the following fields.

Source Settings

- **Input ID:** Enter a unique name to identify this source definition.
- **Brokers:** List of Event Hub 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.
- **Group ID:** Specifies the name of the consumer group to which this Cribl LogStream instance belongs. Should always be `$Default` for Event Hub.
- **From Beginning:** Whether to start reading from the earliest available data. Relevant only during initial subscription. Defaults to `Yes` .

Authentication

- **Enabled:** Defaults to `No` . When toggled to `Yes` :
- **SASL Mechanism:** SASL (Simple Authentication and Security Layer) authentication mechanism to use. Currently, `PLAIN` is the only mechanism supported for Event Hub Kafka brokers.
- **Username:** The username for authentication. For Event Hub, this should always be `$ConnectionString` .
- **Password:** The primary or secondary shared access key from the Event Hub workspace.

TLS Settings (Client Side)

- **Enabled:** Defaults to `Yes` .
- **Validate Server Certs:** Defaults to `No` . (For Event Hub, this should always be false.)

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

 Updated 25 days ago

Metrics

Cribl LogStream supports receiving metrics in these wire formats/protocols: StatsD, StatsD Extended, and Graphite. Automatic protocol detection will happen on the first line received over a TCP connection or a UDP packet. Lines not matching the detected protocol will be dropped.

 Type: **Push** | TLS Support: **No**

Configuring Cribl LogStream to receive metrics.

While on the **Sources** screen, select **Metrics** from the left menu, then click **Add New**. The resulting **New Metrics source** pane provides the following fields.

Source 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.

Advanced Settings

- **Enable Proxy Protocol:** Defaults to `No`. Toggle to `Yes` if the connection is proxied by a device that supports Proxy Protocol V1 or V2.
- **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. Defaults to `1000`.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.

- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__srcIpPort
__metricsInType

Metric Event Schema

Metric data is read into the following event schema:

Text

```
_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
....
```

Sufficient information will be placed into a field called `__criblMetric` that these events can be properly serialized out to any metric outputs (independent of the input type).

 Updated 26 days ago

SQS

Cribl LogStream supports receiving events from [Amazon Simple Queuing Service](#).



Type: **Pull** | TLS Support: **YES** (secure API)

Configuring Cribl LogStream to Receive Data from Amazon SQS

While on the **Sources** screen, select **SQS** from the left menu, then click **Add New**. The resulting **New SQS source** pane provides the following fields.

Source 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. When a non-AWS URL is specified, format must be: {url}/. E.g., https://host:port/.
- **Queue Type:** The queue type used (or created). Defaults to `Standard`.
- **Create Queue:** Create queue if it does not exist.
- **API Key:** If not present, will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM credentials.
- **Secret Key:** If not present, will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.
- **Region:** Region where SQS queue is located. Required.

Advanced Settings

- **AWS Account ID:** SQS queue owner AWS account id. Leave empty if SQS queue is in same AWS account.
- **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`.
- **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`.
- **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 being retrieved by a `ReceiveMessage` request. Defaults to `600`.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

`__inputId`
`__sqsMetadata`


SQS Permissions

The following permissions are needed on the SQS queue:

```
sqs:ListQueues
sqs:SendMessage
sqs:SendMessageBatch
sqs:CreateQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
sqs:GetQueueUrl
```

Troubleshooting Notes





VPC [endpoints](#) for [SQS](#) might need to be set up in your account. Check with your administrator for details.



Updated 19 days ago

S3

Cribl LogStream supports receiving data from [Amazon S3](#), using [event notifications](#) through SQS.

 Type: **Pull** | TLS Support: **YES** (secure API)

The source S3 bucket must be configured to send `s3:ObjectCreated:*` events to an SQS queue, either directly (easiest) or via SNS. See the [event notification configuration guidelines](#) below.

Configuring Cribl LogStream to Receive Data from Amazon S3

While on the **Sources** screen, select **S3** from the left menu, then click **Add New**. The resulting **New S3 source** pane provides the following fields.

Source 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 notifications from. When a non-AWS URL is specified, format must be: {url}/. E.g., `https://host:port/`.
- **Filename Filter:** Regex matching file names to download and process. Defaults to `.*`
- **API Key:** If not present, will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM credentials.
- **Secret Key:** If not present, will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.
- **Region:** Region where the S3 bucket and SQS queue are located. Required.

Assume Role

- **AssumeRole ARN:** Amazon Resource Name (ARN) of the role to assume.
- **External ID:** External ID to use when assuming role.

Advanced Settings

- **AWS Account ID:** SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account.

- **Signature Version:** Signature version to use for signing SQS requests. Defaults to `v4` .
- **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` .
- **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 being retrieved by a `ReceiveMessage` request. Defaults to `600` .

Processing Settings

Custom Command Processor

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:** The amount of time (in milliseconds) that 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. Defaults to `10000` .

Fields (Metadata)

In this section, you can add fields/metadata to each event, using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__inputId
__sqsMetadata

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 the one below: Selecting the queue, and in the **Permissions** tab, click **Edit Policy Document (Advanced)**.
3. In the Amazon S3 console, add a notification configuration to publish events of the `s3:ObjectCreated:*` type to the SQS queue.

SQS Access Policy

```
{
  "Version": "2012-10-17",
  "Id": "example-ID",
  "Statement": [
    {
      "Sid": "example-statement-ID",
      "Effect": "Allow",
      "Principal": {
        "AWS": "*"
      },
      "Action": [
        "SQS:SendMessage"
      ],
      "Resource": "SQS-queue-ARN",
      "Condition": {
        "ArnLike": { "aws:SourceArn": "arn:aws:s3:*:*:bucket-name" }
      }
    }
  ]
}
```

S3 and SQS Permissions

The following permission is required on the S3 bucket:

s3:GetObject

The following permissions are required on the SQS queue:

sqs:ListQueues
sqs:SendMessage
sqs:SendMessageBatch
sqs:CreateQueue
sqs:GetQueueAttributes
sqs:SetQueueAttributes
sqs:GetQueueUrl

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.

 Updated 6 days ago

SNMP Traps

Cribl LogStream supports receiving data from SNMP Traps.

 Type: **Push** | TLS Support: **NO**

Configuring Cribl LogStream to Receive SNMP Traps

While on the **Sources** screen, select **SNMP Traps** from the left menu, then click **Add New**. The resulting **New SNMP Trap source** pane provides the following fields.

Source 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:** UDP port to receive SNMP traps on. Defaults to `162`.

Advanced Settings

- **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. Defaults to `1000`.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__inputId

__snmpVersion : Acceptable values are 0 , 2 , 3 . Versions: 0 =v1, 2 =v2c, 3 =v3.

__srcIpPort : <hostname>|port

__snmpRaw : Buffer containing Raw SNMP packet

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.

 Updated 25 days ago

Datagens

Cribl LogStream supports generating of data from datagen files. See [datagens page](#) for more details.

 Type: **Internal** | TLS Support: **N/A**

Configuring Cribl LogStream to Generate Sample Data

While on the **Sources** screen, select **Datagens** from the left menu, then click **Add New**. The resulting **New Datagen source** pane provides the following fields.

Source 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. Defaults to `10`.

Processing Settings

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

Internal Fields

Cribl LogStream 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(s) for this source:

__inputId



Updated 26 days ago

Cribl Internal

Cribl LogStream allows for capturing and sending its own internal **logs** and **metrics** through routes and pipelines.

 Type: **Internal** | TLS Support: **N/A**

Configuring Cribl LogStream Internal Logs/Metrics to Behave as a Data Source

While on the **Sources** screen, select **Cribl Internal** from the left menu, then toggle **Enable** for CriblLogs and/or CriblMetrics.

CriblLogs Settings

- **Input ID:** Enter a unique name to identify this CriblLogs source definition.

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

CriblMetrics Settings

- **Input ID:** Enter a unique name to identify this CriblMetrics source definition.
- **Metric Name Prefix:** Enter an optional prefix that will be applied to metrics provided by LogStream. The prefix defaults to `cribl.logstream.`

Fields (Metadata)

In this section, you can add fields/metadata to each event using [Eval](#)-like functionality.

- **Name:** Field name.
- **Value:** JavaScript expression to compute field's value (can be a constant).

Conditioning Pipeline

In this section's drop-down list, you can select a single existing pipeline to process data from this input before the data is sent through the routes.

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

Notes

Use these fields to guide these events/metrics through Cribl Routes.

These internal fields are subject to change and modification. Therefore, end users should not rely on them.

 Updated 26 days ago

Collectors

Collectors enable users to dispatch on-demand collection tasks that fetch data from remote locations. As of v.2.2, three collector types are supported: Filesystem, Amazon S3 (or S3-compatible stores), and Script.

How Do Collectors Work

A LogStream node can be configured to retrieve data from a remote system via **Data > Collectors**. Data collection is a multi-step process:

First, define a collector instance. In this step, you configure **collector-specific settings** by selecting a collector type and pointing it at a specific target. (E.g., the target will be a directory if the type is Filesystem, or an S3 bucket/path if the type is Amazon S3.)

Next, Run the collector manually. (Scheduled collection is not yet supported.) In this step, you configure **run-specific settings** – such as the Run Mode, 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 comprised 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 **Pipelines** and **Destination**.

Collectors in Distributed Deployments

In a **distributed deployment**, collectors are set up at the Worker Group level, and the tasks are executed by Worker Nodes. The Master Node oversees the task distribution, and tries to maintain a fair balance across jobs.

When Workers ask for tasks, the Master will normally try to assign the next task from a job with the least tasks in progress. This is known as Least In Flight Scheduling and provides the fairest task distribution for most cases. Default behavior can be changed via **Settings > General Settings > Job Limits > Job Scheduling**.

Collectors

- **Filesystem/NFS** – enables users to collect data from local or remote filesystem locations.
- **S3** – enables users to collect data from Amazon S3 buckets or S3-compatible stores.
- **Script** – enables users to collect data from running custom scripts.

Collector Run Configuration

Once you've added a collector, click **Run** to configure its Mode, Earliest/Latest time range, Filter expression, Log Level, and optional Preview Settings, and to then begin the collector run.

Mode

Depending on your requirements, you can run a collector in any of these modes:

- Preview (default)
- Discovery
- Full Run

Preview

In the default 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.

Discovery

In Discovery mode, a collection job will return only **the list of objects/files** to be collected, but not any 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.

Full Run

In Full Run mode, the collection job is fully executed by Worker Nodes, and will return all data matching the Run configuration.

Earliest and Latest

Optionally, set the start and end date and time for data collection. These are set in your local time.

See time-based tokens below for more.

Filter

This is a JavaScript filter expression evaluated against token values in the provided collector path (see below), and events being collected. The **Filter** value defaults to `true`, which matches all data, but 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` and `.txt` extensions will be fetched and, from those, only those events with host field `myHost` will be collected.

- **Basic Tokens**

Path filtering is supported via token notation in collectors with paths such as [Filesystem](#) or [S3](#). Basic tokens syntax follows that of [JS template literals](#); `${<token_name>}` - where `token_name` is the field (name) of interest.

For example: if 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**

Further filtering is supported via time-based tokens in paths with time partitions. This has a direct effect in earliest and latest boundaries. When a job runs against a path with time partitions, a minimal superset of the required directories to satisfy the time range is traversed before subsequent event `_time` filtering.

Partitions and tokens

- For each path, time partitions must be in decending order. E.g., `dd/mm/yyyy` is not supported.
- For each path, all extracted dates/times are considered in UTC.
- Paths may contain more than one partition. E.g., ``/my/path/2020-04/20/`
- 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

Time-based token syntax follows that of a slightly modified [JS template literals](#);

`${_time: <some_strptime_format_component>}` . Examples:

Filter	Matches
<code>/my/path/\${_time:%Y}/\${_time:%m}/\${_time:%d}/...</code>	<code>/my/path,</code>
<code>/my/path/\${_time:year=%Y}/\${_time:month=%m}/\${_time:date=%d}/...</code>	<code>/my/path,</code>
<code>/my/path/\${_time:%Y-%m-%d}/...</code>	<code>/my/path,</code>

Log Level

Level at which to set task logging. More-verbose levels are useful for troubleshooting jobs and tasks, but use them sparingly.

Preview Settings

In the default Preview mode, you can optionally set these limits:

- **Capture Time (sec):** Amount of time (in seconds) to collect data.
- **Capture Up to N Events:** Maximum number of events to capture.



Updated about 16 hours ago

Filesystem/NFS

Cribl LogStream supports collecting data from a local or a remote filesystem location.

Configuring a Filesystem Collector

While on the **Collectors** screen, click **Add New**. This displays the following options and fields.

- **Collector ID:** Unique ID for this Collector. E.g., `DysonV11Roomba960` .

Collector Settings

The Collector Settings determine how data is collected before processing.

- **Collector Type:** The type of collector to run.



Set this to `Filesystem` to configure the Collector as shown below.

- **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. Symlinks will not be followed. Templating is supported (e.g., `/myDir/${host}/${year}/${month}/`). More on [templates and Filters](#).
- **Recursive:** If set to `Yes` (the default), data collection will recurse through subdirectories.
- **Destructive:** If set to `Yes` , the Collector will delete files after collection. Defaults to `No` .
- **Max Batch Size (files)** : Maximum number of files to batch before recording as results.

Result Settings

The Result Settings determine how LogStream transforms and routes the collected data.

Custom Command Processor

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:** The amount of time (in milliseconds) that 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. Defaults to `10000` .

Fields (Metadata)

In this section, you can add fields/metadata 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), events will be sent to normal routing and event processing. Toggle to `No` to select a specific pipeline/destination combination, in these two additional fields:
 - **Pipeline:** Select a pipeline to process results.
 - **Destination:** Select a destination to receive results.
- **Conditioning Pipeline:** Pipeline to process results before sending to routes. Optional.

Advanced Settings

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` .

Running a Filesystem Collector

To run a Filesystem collector:

1. In the **Actions** column, click **Run**.
2. In the resulting modal, configure the settings for **this** particular job/execution.

For details, see [Collector Run Configuration](#).

3. Click the modal's **Run** button to begin collection.

 Updated a day ago

S3

Cribl LogStream supports collecting data from [Amazon S3](#) stores.

Configuring an S3 Collector

While on the **Collectors** screen, click **Add New**. This displays the following options and fields.

- **Collector ID:** Unique ID for this Collector. E.g., `Attic42TreasureChest` .

Collector Settings

The Collector Settings determine how data is collected before processing.

- **Collector Type:** The type of collector to run.



Set this to `S3` to configure the Collector as shown below.

- **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.

- **Path:** Path, within the bucket, from which to collect data. Templating is supported (e.g., `/myDir/${host}/${year}/${month}/`). More on [templates and Filters](#).
- **API Key:** Enter API key. If empty, will fall back to `env.AWS_ACCESS_KEY_ID` , or to the metadata endpoint for IAM credentials. Optional when running on AWS.
- **Secret Key:** Enter secret key. if empty, will fall back to `env.AWS_SECRET_ACCESS_KEY` , or to the metadata endpoint for IAM credentials. Optional when running on AWS.
- **Region:** S3 Region from which to retrieve data.
- **Recursive:** If set to `Yes` (the default), data collection will recurse through subdirectories.
- **Max Batch Size (files) :** Maximum number of files to batch before recording as results.

Advanced Collector Settings

- **Endpoint:** S3 service endpoint. If empty, LogStream will automatically construct the endpoint from the region.

- **Signature Version:** Signature version to use for signing S3 requests. Defaults to `v4`.
- **AssumeRole ARN:** Amazon Resource Name (ARN) of the role to assume.
- **External ID:** External ID to use when assuming role.

Result Settings

The Result Settings determine how LogStream transforms and routes the collected data.

Custom Command Processor

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:** The amount of time (in milliseconds) that 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. Defaults to `10000`.

Fields (Metadata)

In this section, you can add fields/metadata 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), events will be sent to normal routing and event processing. Toggle to `No` to select a specific pipeline/destination combination, in these two additional fields:
 - **Pipeline:** Select a pipeline to process results.

- **Destination:** Select a destination to receive results.
- **Conditioning Pipeline:** Pipeline to process results before sending to routes. Optional.

Advanced Settings

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 .

Running a Filesystem Collector

To run a S3 collector:

1. In the **Actions** column, click **Run**.
2. In the resulting modal, configure the settings for **this** particular job/execution.

For details, see [Collector Run Configuration](#).

3. Click the modal's **Run** button to begin collection.

 Updated a day ago

Script

Cribl LogStream supports flexible data collection configured by your custom scripts.

Configuring a Script Collector

While on the **Collectors** screen, click **Add New**. This displays the following options and fields.

- **Collector ID:** Unique ID for this Collector. E.g., `sh2GetStuff` .

Collector Settings

The Collector Settings determine how data is collected before processing.

- **Collector Type:** The type of collector to run.



Set this to `Script` to configure the Collector as shown below.

- **Discover Script:** Script to [discover](#) which objects/files to collect. This script should output one task per line in `stdout` .
- **Collect Script:** Script to perform data collections. Pass in tasks from the Discover script as `$CRIBL_COLLECT_ARG` . Should output results to `stdout` .
- **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 LogStream is running. Make sure you understand the impact of what you're executing before you do so! These scripts run as the user running LogStream, so if you are running it as root, these commands will run with root user permissions. 🤖 🤖 🤖

Result Settings

The Result Settings determine how LogStream transforms and routes the collected data.

Custom Command Processor

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:** The amount of time (in milliseconds) that 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. Defaults to `10000`.

Fields (Metadata)

In this section, you can add fields/metadata 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), events will be sent to normal routing and event processing. Toggle to `No` to select a specific pipeline/destination combination, in these two additional fields:
 - **Pipeline:** Select a pipeline to process results.
 - **Destination:** Select a destination to receive results.
- **Conditioning Pipeline:** Pipeline to process results before sending to routes. Optional.

Advanced Settings

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`.

Running a Filesystem Collector

To run a Script collector:

1. In the **Actions** column, click **Run**.
2. In the resulting modal, configure the settings for **this** particular job/execution.

For details, see [Collector Run Configuration](#).

3. Click the modal's **Run** button to begin collection.



Updated 43 minutes ago

Destinations

Cribl LogStream 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, and support backpressure, are referred to as streaming destinations. Supported destinations:

- Splunk
- Splunk Load Balanced
- Splunk HEC
- AWS Kinesis Streams
- AWS CloudWatch Logs
- Elasticsearch
- Honeycomb
- Kafka
- Syslog
- TCP JSON
- Azure Blob Storage
- Azure Event Hubs
- Azure Monitor Logs
- StatsD
- StatsD Extended
- Graphite
- InfluxDB

Non-Streaming Destinations

Destinations that accept events in groups or batches are referred to as non-streaming destinations. Supported destinations:

- S3 Compatible Stores
- Filesystem/NFS

How Does Non-Streaming Delivery Work

Cribl LogStream 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 or copied 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 LogStream will check for any leftover files in progress from prior sessions, and will ensure that they're moved/copied 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 LogStream will enforce the maximum number of open files, by closing files in the order in which they were opened.

Data Delivery

Data is delivered to all destinations on an at-least-once basis. When a Destination is unreachable, there are three possible behaviors:

- **Block** - Cribl LogStream will block incoming events.
- **Drop** - Cribl LogStream will drop events addressed to that Destination.
- **Queue** - Cribl LogStream will Persistent-Queue events to that Destination.

You can configure the desired behavior through a Destination's **Backpressure Behavior** option. If this option is not present, Cribl LogStream's default behavior is to **Block**.

Configuring Destinations

For each destination type, users can create multiple definitions, depending on their requirements. To configure destinations, select **Data > Destinations**, select the desired type from the resulting tiles or left menu, then click **+ Add New**.

 Updated 2 days ago

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.

Configuring Cribl LogStream to Output to an Output Router

While on the **Destinations** screen, select **Output Router** from the vertical menu, then click **Add New**. The resulting **New Router destination** pane contains the following fields.

- **Router Name:** Enter a unique name to identify this router definition.
- **Rules:** A list of event routing rules.
 - **Filter Expression:** JavaScript expression to select events to send to output.
 - **Output:** Output where to send matching events.
 - **Final:** Flag to control whether to stop the event from being checked against other rules. Defaults to `Yes`.

Notes

- An Output Router cannot reference another. This is by design, so as to avoid cycles.
- **Events that do not match any of the rules are dropped.** Use a catchall rule to change this behavior.
- No conditioning can be done here. Use Conditioning Pipelines at Source tier.
- Data can be cloned by turning the `Final` flag to `No`. (The default is `Yes`, i.e., no cloning.)

Example

Scenario:

- Send all events where `host` starts with `69` 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	Output	Final

<code>host.startsWith('69')</code>	San Francisco	Yes
<code>method=='POST' method=='GET'</code>	Seattle	No
<code>method=='POST' method=='GET'</code>	Los Angeles	Yes
<code>true</code>	New York	Yes

 Updated 26 days ago

Splunk Single Instance

Splunk Enterprise is a streaming destination type.

Configuring Cribl LogStream to Output to Splunk Destinations

While on the **Destinations** screen, select **Splunk** from the left menu, then click **Add New**. The resulting **New Splunk Single Instance destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Splunk destination definition.
- **Address:** Hostname of the Splunk receiver.
- **Port:** The port number on the host.
- **Nested field serialization:** Specifies how to serialize nested fields into index-time fields. Defaults to `None`.
- **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 throttle engaged, excesses data will be dropped only if Backpressure Behavior is set to drop, and blocked for all other settings.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

TLS Settings (Client Side)

- **Enabled** defaults to `No`. When toggled to `Yes`:
- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

 **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` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending the data out using this output.

Notes about Forwarding to Splunk

- If events have a Cribl LogStream 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.

 Updated 24 days ago

Splunk Load Balanced

Splunk is a streaming destination type, and with **Splunk Load Balanced** output, you can load balance data out to multiple Splunk receivers.

How Does Load Balancing Work

Cribl LogStream will attempt to load balance outbound data as fairly as possible across all receivers. Data is sent to all receivers simultaneously and the amount sent to each depends on these parameters:

1. Respective destination **weight**
2. Respective destination **historical data**

By default, historical data is tracked for 300s and it is used to influence the traffic sent to each destination so as to ensure that differences decay over time and 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 amount of data. Suppose further that the load balance stats period is set at 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	<i>time=0s ---> time=300s</i>	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	<i>time=300s ---> time=600s</i>	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. A will start the interval with **60** and B with **40**. To determine how many events A and B receive during this interval, Cribl LogStream will use their weights and their stats as follows:

Total number 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: $150 - 40 = 110$

End of interval 2 totals: $A=120+90=210$, $B=80+110=190$, a difference of **20 events** and a ratio of **1.1:1**.

Over the subsequent intervals, the difference becomes exponentially less pronounced and insignificant and thus the load gets balanced fairly.

Configuring Cribl LogStream to Load Balance to Multiple Splunk Destinations

While on the **Destinations** screen, select **Splunk Load Balanced** from the left menu, then click **Add New**. The resulting **New Splunk Load Balanced destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Splunk LB destination definition.
- **DNS Resolution Period (seconds):** Re-resolve any hostnames after each interval of this many seconds, and pick up destinations from A records. Defaults to 60s.
- **Exclude Current Host IPs:** Exclude all IPs of the current host from the list of any resolved hostnames. Defaults to Yes .
- **Load Balance Stats Period (seconds):** Lookback traffic history period. Defaults to 300s. (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 LogStream load balancing, IP settings take priority over those from hostnames.)
- **Indexer Discovery:** Automatically discover indexers in indexer clustering environment. Defaults to No . When toggled to Yes :
 - **Site:** Clustering site from which indexers need to be discovered. In the case of a single site cluster, default is the default entry.
 - **Cluster Master URI:** Full URI of Splunk Cluster Master (scheme://host:port).
 - **Auth Token:** Authentication token required to authenticate to cluster master for indexer discovery.
 - **Refresh Period:** Time interval (in seconds) between two consecutive fetches of indexer list from cluster master. Defaults to 60 .



To enable token authentication on Cluster Master, follow the steps in [this Splunk documentation](#).

- **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 throttle

engaged, excesses data will be dropped only if Backpressure Behavior is set to drop, and blocked for all other settings.

- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.
- **Destinations:** Set of Splunk receivers on which to load balance data.
 - **Host:** 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.
 - **Weight:** The weight to apply to this destination for load balancing purposes.

TLS Settings (Client Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. Defaults to `No`.
- **Server Name (SNI):** Server Name Indication.
- **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 (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`.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending the data out using this output.

SSL Configuration for Splunk Cloud – Special Note

To connect to Splunk Cloud, you **might** need to extract the private and public key from the Splunk-provided Splunk Cloud Certificate (typically bundled in an app), as follows:

Step 1. Test connectivity to Splunk Cloud, using the Root CA certificate:

```
openssl s_client -CApath path_to_ca.pem -connect hostnameToSplunkCloud:9997
```

Step 2. Extract the Private key from the Splunk Cloud Certificate. At the prompt, you will need the `sslPassword` value in `outputs.conf` bundled with the Splunk Cloud app:

```
openssl ec -in path_to_server_cert.pem -out private.pem
```

Step 3. Extract the Public Key for the Server Certificate:

```
openssl x509 -in path_to_server_cert.pem -out server.pem
```

Step 4. In LogStream's Destination TLS section, enter the following:

- **CA Certificate Path:** Path to CA Certificate
- **Private Key Path (mutual auth):** Path to `private.pem` (above)
- **Certificate Path (mutual auth):** Path to `server.pem` (above)

Notes About Forwarding to Splunk

- If events have a LogStream 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.

 Updated 6 days ago

Splunk HEC

Splunk HEC is a streaming destination type. In a typical deployment, Cribl LogStream will be installed/co-located in a Splunk heavy forwarder. If this output is enabled, it can send data out to a Splunk [HEC](#) destination through the [event endpoint](#).

Configuring Cribl LogStream to Output to **Splunk HEC** Destinations

While on the **Destinations** screen, select **Splunk HEC** from the left menu, then click **Add New**. The resulting **New Splunk HEC destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Splunk HEC destination definition.
- **Splunk HEC Endpoint:** URL of a Splunk HEC endpoint to send events to (e.g., `http://myhost.example.com:8088/services/collector/event`).
- **HEC Auth Token:** Splunk HEC authentication token.
- **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 further route this data to another destination.

- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **Request Concurrency:** Maximum number of ongoing requests before blocking. Defaults to `5`.
- **Max body size (KB):** Maximum size, in KB, of the request body. Defaults to `4096`.
- **Flush period (sec):** Maximum time between requests. This could cause the payload size to be smaller than max. Defaults to `1`.
- **Extra HTTP Headers:** Name/Value pairs to pass as additional HTTP headers.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending the data out using this output.

Notes on HTTP-based Outputs

- Cribl LogStream 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 next request.
- When resolving the destination's hostname, LogStream will pick the first IP in the list for use in the next connection. Round-robin DNS would help with event balancing.

 Updated 6 days ago

S3 Compatible Stores

S3 is a non-streaming destination type. Cribl LogStream 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.

Configuring Cribl LogStream to Output to S3 Destinations

While on the **Destinations** screen, select **S3** from the left menu, then click **Add New**. The resulting **New S3 destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this S3 destination definition.
- **S3 Bucket Name:** Name of the destination S3 Bucket.
- **Region:** Region where the S3 bucket is located.
- **API Key:** Enter your AWS API Key. If left blank, LogStream will fall back to `env.AWS_ACCESS_KEY_ID` , or to the metadata endpoint for IAM credentials.
- **Secret Key:** Enter your AWS Secret Key. If left blank, Cribl LogStream will fall back to `env.AWS_SECRET_ACCESS_KEY` , or to the metadata endpoint for IAM credentials.
- **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.
- **Key Prefix:** Prefix to add to files before uploading.
- **Partitioning Expression:** JavaScript expression to define how files are partitioned and organized. If left blank, Cribl LogStream will fall back to `event.__partition` . Defaults to ``${host}/${sourcetype}`` . Partitioning by time is also possible, e.g., ``${host}/${C.Time.strftime(_time, '%Y-%m-%d')}/${sourcetype}``
- **Data Format:** Format of the output data. Defaults to `json` .
- **File Name Prefix:** The output filename prefix. Defaults to `CriblOut` .
- **Compress:** Select the data compression format to use before moving data to final destination. Defaults to `none` . Cribl recommends setting this to `gzip` .
- **Backpressure Behavior:** Select whether to block or drop events when all receivers in this group are exerting backpressure. Defaults to `Block` .

Advanced Settings

- **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` .

- **Server Side Encryption:** Server side encryption type for uploaded objects. Defaults to `none` .
- **Signature Version:** Signature version to use for signing S3 requests. Defaults to `v4` .
- **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 LogStream will close files when **either** of the `Max File Size (MB)` or the `Max File Open Time (Sec)` conditions are met.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Amazon S3 Permissions

The following permissions are needed to write to an Amazon S3 bucket:

```
s3:GetObject
s3:ListBucket
s3:GetBucketLocation
s3:PutObject
```

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

```
__partition
```



Updated 6 days ago

Kinesis Streams

Cribl LogStream can output events to **Amazon Kinesis Data Streams** records of up to 1MB uncompressed. Cribl LogStream does **not** have to run on AWS in order to deliver data to a Kinesis Data Stream.

Configuring Cribl LogStream to Output to Amazon Kinesis Data Streams

While on the **Destinations** screen, select **Kinesis** from the left menu, then click **Add New**. The resulting **New Kinesis destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Kinesis destination definition.
- **Stream Name:** Enter the name of the Kinesis Data Stream to which to send events.
- **API Key:** Enter your AWS API Key. If left blank, LogStream will fall back to `env.AWS_ACCESS_KEY_ID` , or to the metadata endpoint for IAM credentials.
- **Secret Key:** Enter your AWS Secret Key. If left blank, LogStream will fall back to `env.AWS_SECRET_ACCESS_KEY` , or to the metadata endpoint for IAM credentials.
- **Region:** Select the AWS Region where the Kinesis Data Stream is located.
- **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` .
- **Maximum 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.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

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. Connect:  
{"_raw":"07-03-2018 18:33:51.136 -0700 INFO  TcpOutputProc - Connection to 127.  
...
```

 Updated 10 days ago

CloudWatch Logs

Cribl LogStream supports sending data to [Amazon CloudWatch Logs](#). This is a streaming destination type. Cribl LogStream does **not** have to run on AWS in order to deliver data to CloudWatch Logs.

Configuring Cribl LogStream to Output to Amazon CloudWatch Logs

While on the **Destinations** screen, select **CloudWatch Logs** from the left menu, then click **Add New**. The resulting **New CloudWatch Logs destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this CloudWatch destination 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 LogStream instance. (E.g., `myStream_myHost_myOutputId`.)
- **API Key:** Enter your AWS API Key. If left blank, LogStream will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM credentials.
- **Secret Key:** Enter your AWS Secret Key. If left blank, LogStream will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.
- **Region:** AWS region where the CloudWatch Logs group is located.
- **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`.
- **Maximum 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.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

 Updated 10 days ago

SQS

Cribl LogStream supports sending events to [Amazon Simple Queuing Service](#).

Configuring Cribl LogStream to Send Data to Amazon SQS

While on the **Destinations** screen, select **SQS** from the left menu, then click **Add New**. The resulting **New SQS destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this SQS destination.
- **Queue Name:** The name of the AWS SQS queue to send events to.
- **Queue Type:** The queue type used (or created). Defaults to `Standard`.
- **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`.
- **API Key:** Enter your AWS API Key. If left blank, LogStream will fall back to `env.AWS_ACCESS_KEY_ID`, or to the metadata endpoint for IAM credentials.
- **Secret Key:** Enter your AWS Secret Key. If left blank, LogStream will fall back to `env.AWS_SECRET_ACCESS_KEY`, or to the metadata endpoint for IAM credentials.
- **Region:** Region where SQS queue is located.
 - **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **AWS Account ID:** SQS queue owner's AWS account ID. Leave empty if the SQS queue is in the same AWS account where this Cribl LogStream instance is located.
- **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 .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

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 LogStream 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(s) for this Destination:

```
__messageGroupId  
__sqsMsgAttrs  
__sqsSysAttrs
```

 Updated 25 days ago

Filesystem/NFS

Filesystem is a non-streaming destination type that Cribl LogStream can use to output files to a local or a network-attached filesystem (NFS).

Configuring Cribl LogStream to Output to Filesystem Destinations

While on the **Destinations** screen, select **Filesystem** from the left menu, then click **Add New**. The resulting **New Filesystem destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Filesystem destination definition.
- **Output Location:** Final destination for the output files.
- **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.
- **Partitioning Expression:** JavaScript expression to define how files are partitioned and organized. Defaults to ``${host}/${sourcetype}`` . If left blank, Cribl LogStream will fall back to `event.__partition` . Partitioning by time is also possible, e.g.:
``${host}/${C.Time.strftime(_time, '%Y-%m-%d')}/${sourcetype}``
- **Data Format:** Format of the output data. Defaults to `json` .
- **File Name Prefix:** The output filename prefix. Defaults to `CriblOut`
- **Compress:** Data compression format used before moving to final destination. Default `none` . It is recommended that `gzip` is used.
- **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 LogStream will close files when **either** of the `Max File Size (MB)` or the `Max File Open Time (Sec)` conditions are met.

- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

__partition

 Updated 6 days ago

Elasticsearch

Cribl LogStream can send events to an [Elasticsearch](#) cluster using the [Bulk API](#).

Configuring Cribl LogStream to Output to Elasticsearch

While on the **Destinations** screen, select **Elasticsearch** from the left menu, then click **Add New**. The resulting **New Elastic destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Elasticsearch destination definition.
- **Bulk API URL:** URL of an Elasticsearch cluster to send events to. (E.g., `http://<myElasticCluster>:9200/_bulk`.)
- **Index:** Elasticsearch Index where to send events to. Note that this value can be overwritten by an event's `__index` field.
- **Type:** Specify document type to use for events. Note that this value can be overwritten by an event's `__type` field.
- **Authentication Enabled:** Set to `No` by default. Toggle to `Yes` to enter a **Username** and **Password**.
- **Backpressure Behavior:** Specify whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **Request Concurrency:** Maximum number of ongoing requests before blocking. 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.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Field Normalization

This destination normalizes the following fields:

- `_time` becomes `@timestamp` at millisecond resolution.

- `host.name` is set to `host` .

See also our [Elasticsearch Source](#) documentation's **Field Normalization** section.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

`__id`
`__type`
`__index`

Notes on HTTP-based Outputs

- Cribl LogStream 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, LogStream will pick the first IP in the list for use in the next connection. Round-robin DNS would help with event balancing.

 Updated 23 days ago

Honeycomb

Cribl LogStream supports sending events to a [Honeycomb](#) dataset.

Configuring Cribl LogStream to Output to Honeycomb

While on the **Destinations** screen, select **Honeycomb** from the left menu, then click **Add New**. The resulting **New Honeycomb destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this Honeycomb destination definition.
- **Dataset name:** Name of the dataset to send events to. (E.g., `iLoveObservabilityDataset`.)
- **API Key:** Team API Key to which the dataset belongs. (E.g., `teamWilde`.)
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **Request Concurrency:** Maximum number of ongoing requests before blocking. Defaults to `5`.
- **Max Body Size (KB):** Maximum size of the request body. Defaults to `4096` KB.
- **Flush Period (sec):** Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.
- **Extra HTTP Headers:** Name/Value pairs to pass as additional HTTP headers.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Notes on HTTP-based Outputs

- Cribl LogStream 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, LogStream will pick the first IP in the list for use in the next connection. Round-robin DNS would help with event balancing.

 Updated 25 days ago

TCP JSON

Cribl LogStream supports sending data over TCP in JSON format. **TCP JSON** is a streaming destination type.

Configuring Cribl LogStream to Output in TCP JSON Format

While on the **Destinations** screen, select **TCP JSON** from the left menu, then click **Add New**. The resulting **New TCP JSON destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **Host:** Hostname of the receiver.
- **Port:** Port number to connect to on the host.
- **Auth Token:** Optional authentication token to include as part of the connection header. Defaults to empty.
- **Compression:** Codec to use to compress the data before sending. Defaults to `None`.
- **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 throttle engaged, excesses data will be dropped only if Backpressure Behavior is set to drop, and blocked for all other settings.
- **Backpressure Behavior:** Specifies whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

TLS Settings (Client Side)

- **Enabled:** Defaults to `No`. When toggled to `Yes`:
- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

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` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Format

TCP JSON events are sent in [newline-delimited JSON](#) format:

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 here](#).



Updated 24 days ago

Syslog

Cribl LogStream supports sending of data over syslog via TCP. **Syslog** is a streaming destination type.

Configuring Cribl LogStream to output in **Syslog** format

While on the **Destinations** screen, select **Syslog** from the left menu, then click **Add New**. The resulting **New Syslog destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **Protocol:** The network protocol to use for sending out syslog messages. Defaults to TCP .
- **Address:** Address/hostname of the receiver.
- **Port:** Port number to connect to on the host.
- **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. Multiple byte units such as KB, MB, GB etc. are also allowed. E.g., 42 MB. Default value of 0 indicates no throttling. When throttle engaged, excesses data will be dropped only if Backpressure Behavior is set to drop, and blocked for all other settings.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block` .

TLS Settings (Client Side)

- **Enabled:** Defaults to `No` . When toggled to `Yes` :
- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

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` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

```
__priority
__facility
__severity
__procid
__appname
__msgid
__syslogout
```

 Updated 6 days ago

Kafka

Cribl LogStream supports sending data to a [Kafka](#) topic. **Kafka** is a streaming destination type.

Configuring Cribl LogStream to Output to Kafka

While on the **Destinations** screen, select **Kafka** from the left menu, then click **Add New**. The resulting **New Kafka destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination 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 `__topic` field.
- **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. Defaults to `Gzip`.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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:** Select the SASL (Simple Authentication and Security Layer) authentication mechanism to use,
- **Username:** The username for authentication.
- **Password:** The password for authentication.

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`:
- **Schema Registry URL:** URL for access to the Confluent Schema Registry. (E.g., `http://<hostname>:8081`.)

- **Default Key Schema ID:** Used when `__keySchemaId0ut` is not present to transform key values. Leave blank if key transformation is not required by default.
- **Default Value Schema ID:** Used when `__valueSchemaId0ut` 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:



These have the same format as the [TLS Settings \(Client Side\)](#) below.

TLS Settings (Schema Registry)

- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

TLS Settings (Client Side)

- **Validate Server Certs:** Require client to reject connections to servers whose certs are not signed by one of the supplied CAs. 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.

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 .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

__topicOut

__key

__headers

__keySchemaIdOut

__valueSchemaIdOut

 Updated 6 days ago

Azure Blob Storage

[Azure Blob Storage](#) is a non-streaming destination type. Cribl LogStream does **not** have to run on Azure in order to deliver data to it. [Azure Data Lake Storage Gen2](#) (hierarchical namespace) is also supported.

Configuring Cribl LogStream to output to Azure Blob Storage

While on the **Destinations** screen, select **Azure > Azure Blob Storage** from the left menu, then click **Add New**. The resulting **New Blob Storage destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **Account Name:** Enter your Azure Storage Account Name. If left blank, Cribl LogStream will fall back to `env.AZURE_STORAGE_ACCOUNT`.
- **Account Key:** Enter your Azure Storage Key. If left blank, Cribl LogStream will fall back to `env.AZURE_STORAGE_KEY`.
- **Container Name:** Enter the container name. (A container organizes a set of blobs, similar to a directory in a file system.)
- **Create Container:** Defaults to `No`. Toggle to `Yes` to create the configured container in Azure Blob Storage if it does not already exist.
- **Blob Prefix:** Prefix to add to files 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.
- **Partitioning Expression:** JavaScript expression to define how files are partitioned and organized. Defaults to ``${host}/${sourcetype}``. If left blank, Cribl LogStream will fall back to `event.__partition`.
- **Data Format:** Format of the output data. Defaults to `json`.
- **File Name Prefix:** The output filename prefix. Defaults to `CriblOut`.
- **Compress:** Data compression format used before moving to final destination. Defaults to `none`. Cribl recommends setting to `gzip`.
- **Backpressure Behavior:** Whether to block or drop events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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 LogStream will close files when **either** of the Max File Size (MB) or the Max File Open Time (Sec) conditions are met.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

__partition



Updated 6 days ago

Azure Monitor Logs

Cribl LogStream supports sending of data over to [Azure Monitor Logs](#). This is a streaming destination type.

Configuring Cribl LogStream to Output to Azure Monitor Logs

While on the **Destinations** screen, select **Azure | Monitor Logs** from the left menu, then click **Add New**. The resulting **New Monitor Logs destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **Workspace ID:** Enter the Azure Log Analytics Workspace ID. (In the Azure Dashboard, see **Workspace->Advanced settings**.)
- **Workspace Key:** Enter the Azure Log Analytics Workspace Primary or Secondary Shared Key. (In the Azure Dashboard, see **Workspace->Advanced settings**.)
- **Log Type:** The Record Type of events sent to this LogAnalytics workspace. Defaults to `Cribl`.
- **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 in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **Request Concurrency:** Maximum number of ongoing requests before blocking. Defaults to `5`.
- **Max Body Size (KB):** Maximum size of the request body. Defaults to `4096`.
- **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.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Notes on HTTP-based Outputs

- Cribl LogStream 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, LogStream will pick the first IP in the list for use in the next connection. Round-robin DNS would help with event balancing.

 Updated 25 days ago

Azure Event Hubs

Cribl LogStream supports sending data to [Azure Event Hubs](#). This is a streaming destination type.

Configuring Cribl LogStream to Output to Azure Event Hubs

While on the **Destinations** screen, select **Azure | Event Hubs** from the left menu, then click **Add New**. The resulting **New Event Hubs destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination 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.
- **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`.
- **Compression:** Codec to use to compress the data before sending it to Event Hub Kafka brokers. Defaults to `Gzip`.
- **Backpressure Behavior:** Whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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`.
- **Password:** Primary or Secondary shared access key from the Event Hub workspace.

TLS Settings (Client Side)

- **Enabled** Defaults to `Yes`.

- **Validate Server Certs:** Defaults to `No` . For Event Hub, this should always be false.

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` .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

`__topicOut`
`__key`
`__headers`
`__keySchemaIdOut`
`__valueSchemaIdOut`

 Updated 25 days ago

StatsD

Cribl LogStream supports sending data to a [StatsD](#) destination. This is a streaming destination type.

Configuring Cribl LogStream to Output via StatsD

While on the **Destinations** screen, select **Metrics | StatsD** from the left menu, then click **Add New**. The resulting **New StatsD destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination 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`.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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`.

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`.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

 Updated 6 days ago

StatsD Extended

Cribl LogStream supports sending data to a [StatsD](#) destination. This is a streaming destination type.

Configuring Cribl LogStream to Output via StatsD Extended

While on the **Destinations** screen, select **Metrics | StatsD Extended** from the left menu, then click **Add New**. The resulting **New StatsD Extended destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination 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`.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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`.

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`.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

 Updated 6 days ago

Graphite

Cribl LogStream supports sending data to a [Graphite](#) backend destination. This is a streaming destination type.

Configuring Cribl LogStream to Output to a Graphite Backend

While on the **Destinations** screen, select **Metrics | Graphite** from the left menu, then click **Add New**. The resulting **New Graphite destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination 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`.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

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`.

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`.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

 Updated 6 days ago

SNMP Traps

Cribl LogStream supports forwarding of SNMP Traps out.

Configuring Cribl LogStream to Forward SNMP Traps

While on the **Destinations** screen, select **SNMP Traps** from the left menu, then click **Add New**. The resulting **New SNMP destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **SNMP Trap Destinations:* One or more SNMP destinations to forward traps to.
 - **Address:** Destination host
 - **Port:** Destination port. Defaults to 162 .

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

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.

 Updated 25 days ago

InfluxDB

Cribl LogStream supports sending data to [InfluxDB](#).

Configuring Cribl LogStream to Output to InfluxDB

While on the **Destinations** screen, select **InfluxDB** from the left menu, then click **Add New**. The resulting **New InfluxDB destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this InfluxDB destination definition.
- **Write API URL:** URL of an InfluxDB cluster where to send events to. (E.g., `http://localhost:8086/write`.)
- **Database Name:** The database on which to write data points.
- **Timestamp Precision:** Sets the precision for the supplied Unix time values. Defaults to `Milliseconds`.
- **Dynamic Value Fields:** When enabled, LogStream 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`.
- **Authentication Enabled:** Set to `No` by default. Toggle to `Yes` to enter a **Username** and **Password**.
- **Backpressure Behavior:** Select whether to block, drop, or queue events when all receivers in this group are exerting backpressure. Defaults to `Block`.

Advanced Settings

- **Request Concurrency:** Maximum number of ongoing requests before blocking. Defaults to `5`.
- **Max body size (KB):** Maximum size of the request body. Defaults to `4096` KB.
- **Flush period (sec):** Maximum time between requests. Low values could cause the payload size to be smaller than its configured maximum. Defaults to `1`.
- **Extra HTTP Headers:** Name/Value pairs to pass as additional HTTP headers.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

 Updated 23 days ago

MinIO

MinIO is a non-streaming destination type, to which Cribl LogStream can output objects.

Configuring Cribl LogStream to output to MinIO destinations.

While on the **Destinations** screen, select **MinIO** from the left menu, then click **Add New**. The resulting **New MinIO destination** pane contains the following fields.

- **Output ID:** Enter a unique name to identify this destination definition.
- **MinIO Endpoint:** MinIO service url (e.g. `http://minioHost:9000`).
- **MinIO Bucket Name:** Name of the destination MinIO bucket.
- **API Key:** If left blank, LogStream will fall back to `env.AWS_ACCESS_KEY_ID` , or to the metadata endpoint for IAM credentials.
- **Secret Key:** If left blank, Cribl LogStream will fall back to `env.AWS_SECRET_ACCESS_KEY` , or to the metadata endpoint for IAM credentials.
- **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.
- **Key Prefix:** Prefix to append to files before uploading.
- **Partitioning Expression:** JavaScript expression to define how files are partitioned and organized. If left blank, Cribl LogStream will fall back to `event.__partition` . Defaults to ``${host}/${sourcetype}`` .
- **Data Format:** Format of the output data. Defaults to `json` .
- **File Name Prefix:** The output filename prefix. Defaults to `CriblOut` .
- **Compress:** Select the data compression format to use before moving data to final destination. Defaults to `none` . Cribl recommends setting this to `gzip` .
- **Backpressure Behavior:** Select whether to block or drop events when all receivers in this group are exerting backpressure. Defaults to `Block` .

Advanced Settings

- **Region:** Region where the MinIO service/cluster is located.
- **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` .

- **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 LogStream will close files when **either** of the Max File Size (MB) or the Max File Open Time (Sec) conditions are met.

Conditioning Pipeline

- **Conditioning Pipeline:** Pipeline to process data before sending out the data using this output.

Internal Fields

Cribl LogStream uses a set of internal fields to assist in forwarding data to a destination.

Field(s) for this destination:

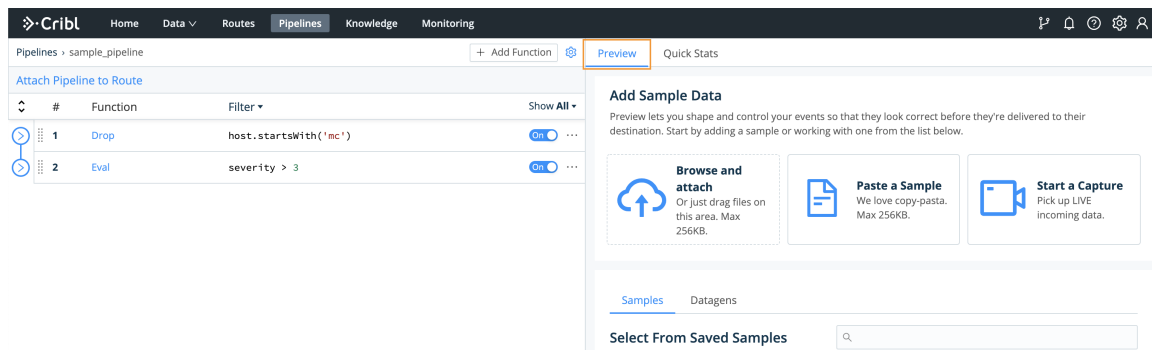
__partition



Updated 6 days ago

Data Preview

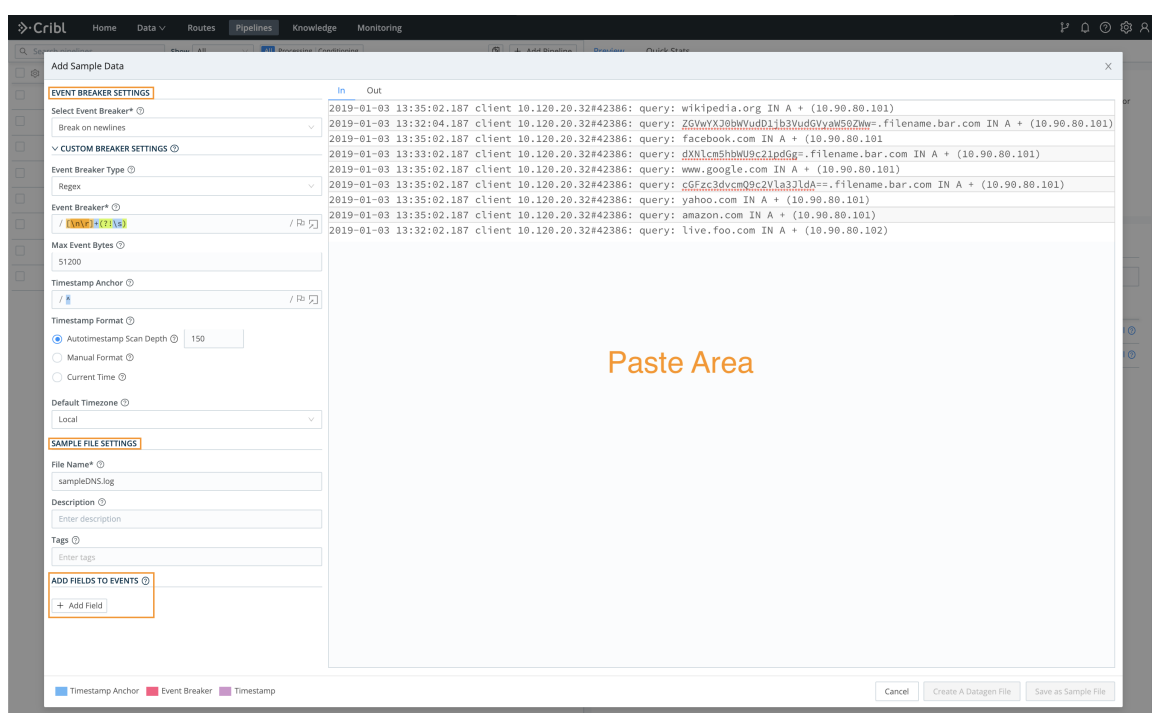
Data Preview is a feature that allows for visual inspection of events as they make their trip into a pipeline. It helps users shape and control events before they're delivered to a destination, as well as assisting with troubleshooting functions. It works by taking a set of Sample events, passing them through the pipeline, and displaying the result on a different pane. Any time a function is modified, added, or removed, the pipeline changes and so does its output.



While you're in a pipeline, you can add samples through one of the supported options: Upload, Paste, or Capture. The Upload and Paste options work with content that needs to be broken into events, while the Capture option works with events only.

Adding Sample Data (Using Paste as an Example)

When you click on the corresponding option, you'll be presented with a screen like the one shown below.





The Capture screen is slightly different – it does not require event breaking.

Paste Area

This is where the content of the paste (or uploaded file) is displayed.

Event Breaker Settings

An event breaker is a regular expression that tells Cribl LogStream how to break the file or paste content into events. Breaking will occur at the **start** of the match. Cribl LogStream 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.

Fields

The Fields section enables users to add, or overwrite, key/value pairs on the sample.

In Tab: Displaying Samples on the Way IN to the Pipeline

There are two display options for the event: Text and Table. (You can also download data as JSON or NDJSON, using the **Advanced Settings** menu at top right.) Each format can be useful, depending on the type of data you are previewing.

The screenshot shows the Cribl LogStream interface. The top navigation bar includes Home, Data, Routes, Pipelines, Knowledge, and Monitoring. The main area is divided into two sections: Pipeline configuration and Event preview.

Pipeline Configuration:

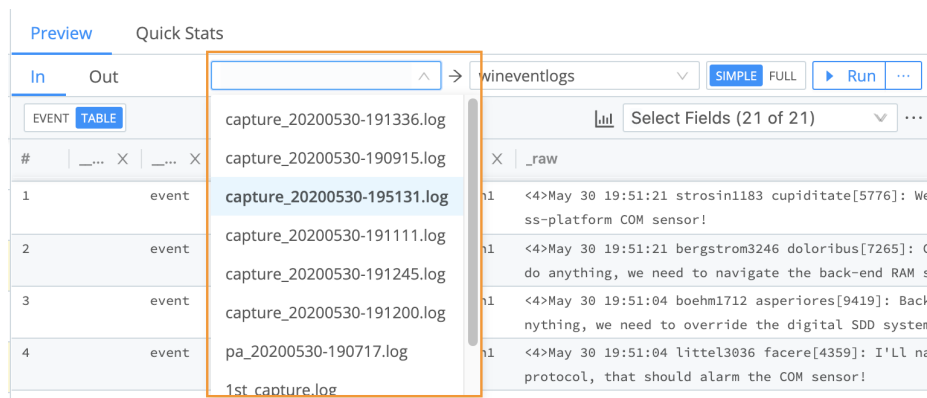
- Attach Pipeline to Route:** capture_20200530-1951...
- Function List:**
 - 1 Eval: `!(sourcetype=="WinEventLog:Security" || source=...`
 - 2 Regex Extract: `true`
 - 3 Comment: Drop Process Start & End emitted by Splunk Universal Forwar...
 - 4 Drop: `(__eventCode=="4688" || __eventCode=="4689") && ...`
 - 5 Comment: Drop Repetitious Token Assignment Events
 - 6 Drop: `__eventCode=="4703" && /Security ID:\s*NT AUTHOR...`
 - 7 Comment: Remove event descriptions
 - 8 Mask: `__eventCode=="4624" || __eventCode=="4688"`
 - 9 Comment: Remove timestamps
 - 10 Mask: `true`

Event Preview (In Tab):

#	Event	Field	Value
1	event	datagen:datagen1	<4>May 30 19:51:21 ss-platform COM si
2	event	datagen:datagen1	<4>May 30 19:51:21 do anything, we ni
3	event	datagen:datagen1	<4>May 30 19:51:21 nything, we need
4	event	datagen:datagen1	<4>May 30 19:51:21 protocol, that sh
5	event	datagen:datagen1	<7>May 30 19:51:1 te the solid state protocol so we can quantify the
6	event	datagen:datagen1	<4>May 30 19:51:14 reichert4779 soluta[8884]: Use you can quantify the haptic driver!
7	event	datagen:datagen1	<4>May 30 19:51:11 lftel13936 facere[4359]: I'll n protocol, that should alarm the COM sensor!
8	event	datagen:datagen1	<4>May 30 19:51:17 pollich1823 et(4117): Hacking th g, we need to back up the cross-platform COM firew
9	event	datagen:datagen1	<4>May 30 19:51:11 lenke7874 ipsun[196]: We need to bus!
10	event	datagen:datagen1	<4>May 30 19:51:11 medhurst7847 repellendus(6460): I monitor, that should protocol the JSON circuit!

On the right side of the preview, there are options to **Show Dropped Events**, **Show Internal Fields**, **Enable Diff**, and **Enable CPU profiling**. At the bottom right, there are links to **Download as JSON**, **Download as NDJSON**, **Preview Log**, and **Show Profiler**.

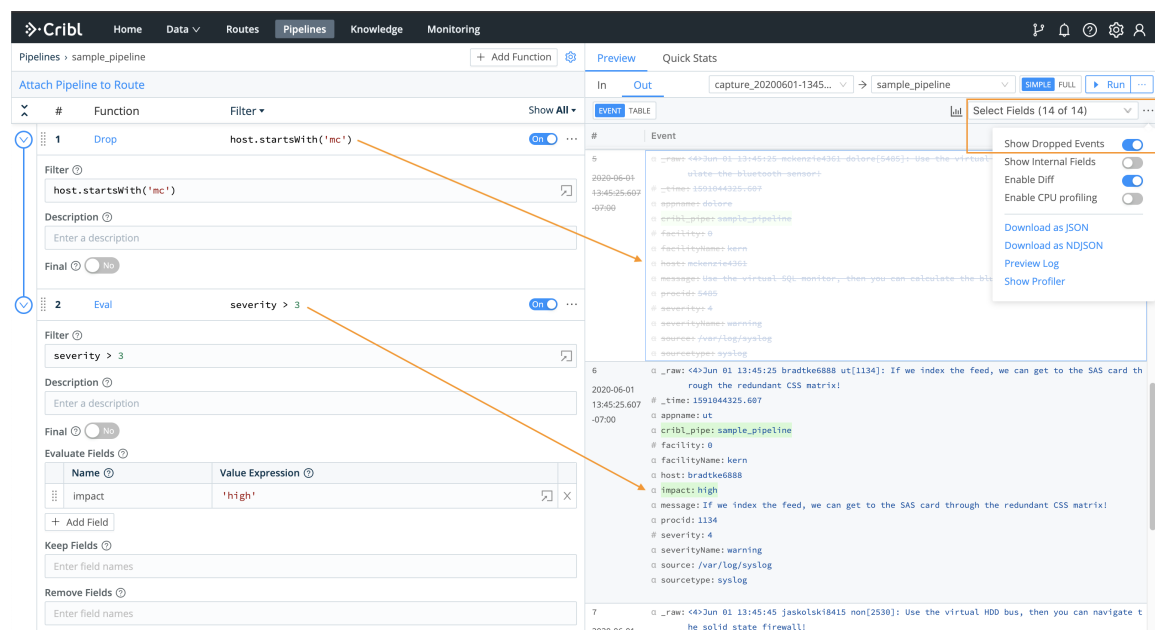
As you add more samples to your system, you can easily access them via the Samples drop-down near the top right.



Out Tab: Displaying Samples on the Way OUT of the Pipeline

As data traverses functions in a pipeline, events can be modified, and some might be dropped altogether. When they're 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.

When LogStream's processing adds new fields, these fields are highlighted green. You can control these fields' display using the **Select Fields** drop-down.



Updated 4 days ago

Securing Data

Cribl LogStream can be used to encrypt sensitive data in real-time and route it to an end system. Decrypted retrieval can be implemented on a per-system basis. At the time of this writing decryption is supported only when Splunk is the end system.

- Data Encryption
- Data Decryption

 Updated 4 days ago

Encryption

Encryption of Data in Motion

With Cribl LogStream, 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 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 key class

@param - keyId - encryption keyId, takes precedence over keyclass

@param - defaultVal - what to return if encryptions 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 LogStream 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 CLI

When using the `local` key management system, encryption keys in Cribl LogStream 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
```

Sample Command Output

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/cribld keys add --help
```

Sample Command Output

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
```

Adding a key to keyclass 1 with no expiration date:

```
$CRIBL_HOME/bin/cribl keys add -c 1 -i
```

Sample Command Output

Adding key: success. Key count=1

Listing keys to verify key generation:

```
$CRIBL_HOME/bin/cribl keys list
```

Sample Command Output

keyId	algorithm	keyclass	kms	created	expires	useIV
1	aes-256-cbc	1	local	1545243364.342	0	true

Configuring Keys with UI

The key management interface can be accessed through **Settings > 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

[Get Key Bundle](#)

[+ Add New](#)

To protect against accidental changes, key parameters can *only* be modified through configuration files.

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

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. The `cribl.secret` and `keys.json` files in `$CRIBL_HOME/local/cribl/auth` (**in the Cribl instance where encryption happened**) should be synced/copied over to the ones on the Search Head/decrypting side. When using the UI, these files can be downloaded through the **Get Key Bundle** button.

 Updated 4 days ago

Decryption

Decryption of Data

Cribl LogStream currently supports decryption only when Splunk is the end system. In Splunk, decryption is available to users of any role with permissions to the `decrypt` command. Further restrictions can be applied with Splunk capabilities. See below for more.

Decrypting in Splunk

Decryption in Splunk is implemented via a custom command called `decrypt`. 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 `decrypt`.



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 – `decrypt` is scoped to run locally on the installed search head.

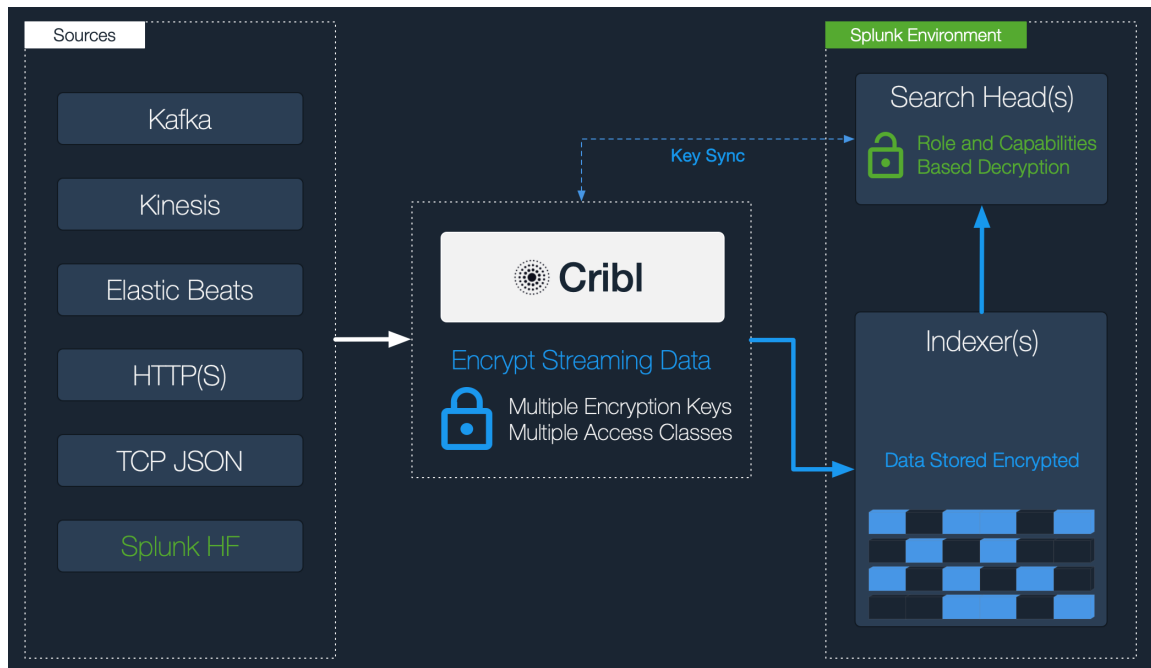
Restricting Access with Splunk Capabilities

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
<code>cribl_keyclass_1</code>	<code>1</code>
<code>cribl_keyclass_2</code>	<code>2</code>
<code>...</code>	<code>...</code>
<code>cribl_keyclass_N</code>	<code>N</code>

Configuring Splunk Search Head to Decrypt Data

You set up decryption in Splunk according to this schematic:



1. Install the Cribl App for Splunk on your Search Head. As of 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 to search head mode with the command: `$CRIBL_HOME/bin/cribld mode=searchhead`. When installed as a Splunk app, `$CRIBL_HOME` is `$SPLUNK_HOME/etc/apps/cribl`.
2. Assign [permissions](#) to the `decrypt` command, per your requirements.
3. Assign [capabilities](#) to your roles, per your requirements. If you'd like to create more capabilities, ensure that they follow the naming convention defined above.
4. Sync `auth/(cribl.secret|keys.json)`. To successfully decrypt data, the `decrypt` command will need access to the same keys that were used to [encrypt](#). The `cribl.secret` and `keys.json` files in `$CRIBL_HOME/local/cribl/auth` (**in the Cribl instance where encryption happened**) should be synced/copied over to the files on the Search Head/decrypting side. When using the UI, these files can be downloaded through the **Get Key Bundle** button.

 Updated 4 days ago

Scripts

Admins can run scripts (e.g., shell scripts) from within Cribl LogStream by configuring and executing them thru **Settings > Scripts**. They are typically used to call custom automation jobs or in general trigger tasks on demand. For example, you can use Scripts to run an Ansible job, or place a call to another automation system, when Cribl LogStream configs are updated.

⚠ With Great Power Comes Great Responsibility!

Scripts will allow you to execute almost anything on the system where Cribl LogStream is running. Make sure you understand the impact of what you're executing before you do so!

Manage Scripts + Add New

⚠ 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*

Command*

Description

Arguments

Env Variables ×

+ Add Variable

Run Delete 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

i Scripts in Distributed Deployments

- Scripts can be deployed from Master Node but can only be **Run** locally from each Worker Node.
- If the Script command is referencing a file, e.g., `420.sh` it has to exist on the Cribl LogStream instance. I.e. the Script management interface cannot be used to upload or manage script files.

📅 Updated 4 days ago

Datagens

Data generators for testing and troubleshooting


Cribl LogStream'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 [samples](#) files or live [captures](#).


[Preview](#) [Stats](#)

Add Sample Data

Preview lets you shape and control your events so that they look correct before they're delivered to their destination. Start by adding a sample or working with one from the list below.


**Browse and attach**
Or just drag files on this area. Max 256KB.

**Paste a Sample**
We love copy-pasta. Max 256KB.

**Start a Capture**
Pick up LIVE incoming data.

[Samples](#) [Datagens](#)

Select From Saved Datagens

<input type="checkbox"/>  File Name	Created ↑	Expires	Size	Events	Actions
<input type="checkbox"/> > weblog.log	2019-11-26 14:39:08	Never	42.40KB	100	Setup Datagen
<input type="checkbox"/> > syslog.log	2019-11-26 14:39:08	Never	19.97KB	100	Setup Datagen
<input type="checkbox"/> > apache_common.log	2019-11-26 14:39:08	Never	17.12KB	100	Setup Datagen
<input type="checkbox"/> > apache_error.log	2019-11-26 14:39:08	Never	23.32KB	100	Setup Datagen

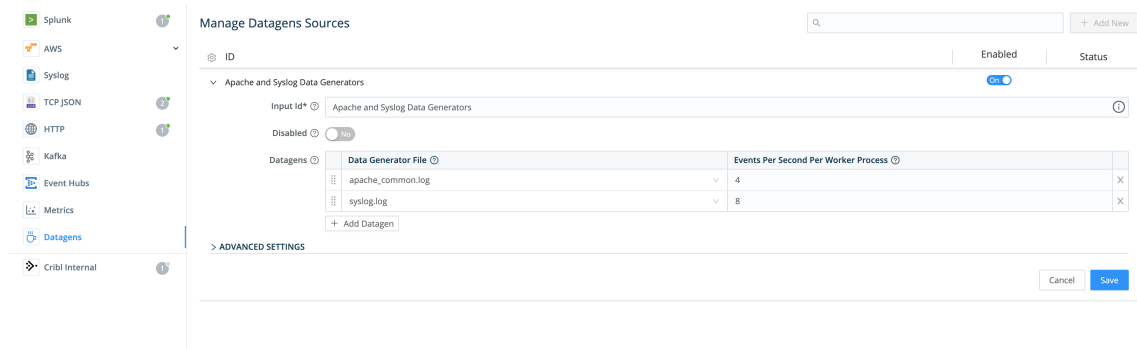
As outlined in the following tutorial: Once you've created a template, you can configure a Datagen source to use it to generate real-time data at a given EPS (events per second) rate.

Enabling a Datagen

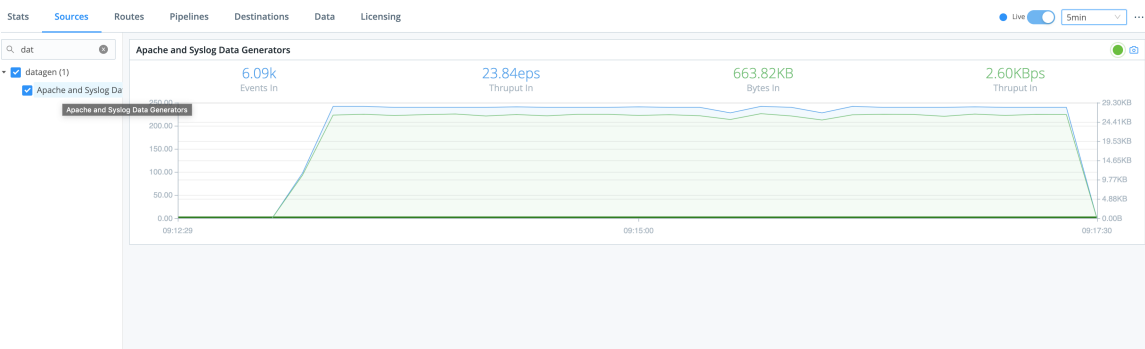
To see how Datagens work, start by enabling a pair of LogStream's out-of-the-box generators:

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 (e.g., `syslog.log`) and set it at 8 EPS/worker process. Hit **Save**.



In the **Monitoring** screen, under **Sources**, search for `datagen` and confirm that the Source is generating data.



Creating a Datagen Template from a Sample File

To convert a sample into a template:

Go to **Preview > Paste a Sample**, and add a sample like the [AWS VPC Flow logs](#) below:

Sample VPC Flow Logs

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418!
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 14185:
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 :
2 123456789010 eni-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 :
2 123456789010 eni-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:12:
```

From the **Event Breaker** dropdown, 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).

Once you've verified these results, click **Create a Datagen File**.

Add Sample Data

EVENT BREAKER SETTINGS

Select Event Breaker*

AWS VPC Flow

> CUSTOM BREAKER SETTINGS

SAMPLE FILE SETTINGS

File Name*

sample_20200114-104732.log

Description

Enter description

Tags

Enter tags

ADD FIELDS TO EVENTS

+ Add Field

In Out

```

2 123456789010 ent1-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418538078 1418538078 ACCEPT OK
2 123456789010 ent1-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418538078 1418538078 REJECT OK
2 123456789010 ent1-1a2b3c4d - - - - - 1431280876 1431280934 - NODATA
2 123456789010 ent1-4b118871 - - - - - 1431280876 1431280934 - SKIPDATA
2 123456789010 ent1-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027 1432917142 ACCEPT OK
2 123456789010 ent1-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 1432917142 REJECT OK
2 123456789010 ent1-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:1234:a102:3384:8879:34cf:4071 34892 22 6 54 8855 1477913788 1477913828 ACCEPT OK

```

Timestamp Anchor Event Breaker Timestamp

Cancel Create A Datagen File Save as Sample File

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 current time – in each newly generated event – using the given format. In this case, `%s` is the desired strptime format for the timestamp (i.e. 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

`%(asctime)s` / `%s`

Default Timezone

UTC

Timestamp Format

☐ Autotimestamp Scan Depth

☒ Manual Format

☐ Current Time

EVENT TABLE

Select Fields (2 of 2)

#	Event
1	2 123456789010 ent1-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418538078 1418538078 ACCEPT OK 2014-12-13 23:06:50.00 0-05:00
2	2 123456789010 ent1-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 1418538078 1418538078 REJECT OK 2014-12-13 23:06:50.00 0-05:00
3	2 123456789010 ent1-1a2b3c4d - - - - - 1431280876 1431280934 - NODATA 2015-05-10 14:01:16.00 0-04:00
4	2 123456789010 ent1-4b118871 - - - - - 1431280876 1431280934 - SKIPDATA 2015-05-10 14:01:16.00 0-04:00
5	2 123456789010 ent1-1235b8ca 203.0.113.12 172.31.16.139 0 0 1 4 336 1432917027 1432917142 ACCEPT OK 2015-05-29 12:30:27.00 0-04:00
6	2 123456789010 ent1-1235b8ca 172.31.16.139 203.0.113.12 0 0 1 4 336 1432917094 1432917142 REJECT OK 2015-05-29 12:31:34.00 0-04:00
7	2 123456789010 ent1-f41c42bf 2001:db8:1234:a100:8d6e:3477:df66:f105 2001:db8:1234:a102:3384:8879:34cf:4071 34892 22 6 54 8855 1477913788 1477913828 ACCEPT OK 2016-10-31 07:55:08.00 0-04:00


Cancel Save as Datagen File


To confirm that the Datagen file has been created, check **Preview > Datagens**.


[Preview](#) [Stats](#)

Add Sample Data

Preview lets you shape and control your events so that they look correct before they're delivered to their destination. Start by adding a sample or working with one from the list below.

**Browse and attach**
Or just drag files on this area. Max 256KB.

**Paste a Sample**
We love copy-pasta. Max 256KB.

**Start a Capture**
Pick up LIVE incoming data.

[Samples](#) [Datagens](#)

Select From Saved Datagens

<input type="checkbox"/>	File Name	Created ↑	Expires	Size	Events	Actions
<input type="checkbox"/>	> vpc-flow-datagen.log	2020-01-14 11:00:26	Never	960.00B	7	Setup Datagen
<input type="checkbox"/>	> weblog.log	2019-11-26 14:39:08	Never	42.40KB	100	Setup Datagen
<input type="checkbox"/>	> syslog.log	2019-11-26 14:39:08	Never	19.97KB	100	Setup Datagen
<input type="checkbox"/>	> apache_common.log	2019-11-26 14:39:08	Never	17.12KB	100	Setup Datagen
<input type="checkbox"/>	> apache_error.log	2019-11-26 14:39:08	Never	23.32KB	100	Setup Datagen

Now, to start using your newly created datagen file, go back to **Sources > Datagens**. Add it using the drop-down shown below.

Splunk

AWS

Syslog

TCP JSON

HTTP

Kafka

Event Hubs

Metrics

Datagens

Cribl Internal

Manage Datagens Sources

☐ ID

Enabled

Status

☐ Apache and Syslog Data Generators

On

Input Id*

Apache and Syslog Data Generators

Disabled

Datagens

Data Generator File	Events Per Second Per Worker Process	
apache_common.log	4	X
syslog.log	8	X
	10	X

+> ADVANCED SETTINGS

Delete Source

Clone Source

Cancel

Save

 Updated 4 days ago

CLI Reference

Command line interface basics

Beyond starting and stopping the Cribl LogStream server, LogStream'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>
```

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>`.

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.

help

Displays help (commands list).

```
Cribl LogStream - N.n.n-<build no.>
Usage: [sub-command] [options] [args]
```

Commands:

help	- Display help
reload	- Reload Cribl LogStream
restart	- Restart Cribl LogStream
start	- Start Cribl LogStream
status	- Status of Cribl LogStream
stop	- Stop Cribl LogStream
version	- Print Cribl LogStream version and installation type

auth	– Cribl LogStream Auth
boot-start	– Enable/Disable Cribl LogStream boot-start
diag	– Manage diagnostics bundles
groups	– Manage worker groups
keys	– Manage encryption keys
mode-searchhead	– Configure Cribl LogStream to run on a Splunk Search Head
nc	– Listen on a port for traffic and output stats and data
node	– Execute a JavaScript file
pipe	– Feed stdin to a pipeline
scope	– Grep your apps by the syscalls
splunk-decrypt	– Splunk decrypt search command
task	– Run Cribl LogStream task
vars	– Manage global variables

reload

Reloads Cribl LogStream. Executes immediately.

```
Reload request submitted to Cribl LogStream
```

restart

Restarts Cribl LogStream. Executes immediately.

```
Stopping Cribl LogStream, process 56572
.....
Cribl LogStream is not running
Starting Cribl LogStream...
..
Cribl LogStream started with pid 57233
API Server is available at http://192.168.0.100:9000
```

start

Starts Cribl LogStream. Executes immediately.

```
Starting Cribl LogStream...
..
Cribl LogStream started with pid 57279
API Server is available at http://192.168.0.100:9000
```

status

Displays status of Cribl LogStream. Executes immediately.

```
Cribl LogStream is running with pid 57279
API Server is available at http://192.168.0.100:9000
```

stop

Stops Cribl LogStream. Executes immediately.

```
Stopping Cribl LogStream, process 57233
.....
Cribl LogStream is not running
```

version

Displays Cribl LogStream version and installation type. Executes immediately.

```
Version: 2.2-0####x##
Installation type: standalone
```

auth

Log into or out of Cribl LogStream.

```
Commands:
login - Log in to Cribl LogStream, args:
  [-h <host>]      - Host URL (e.g. http://localhost:9000)
  [-u <username>]  - Username
  [-p <password>]  - Password
  [-f <file>]      - File with credentials
logout - Log out from Cribl LogStream
```

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 LogStream boot-start.

Usage: [sub-command] [options] [args]

Commands:

disable - Disable Cribl LogStream boot-start, args:

[-m <manager>] - Init manager (systemd|initd)

[-c <configDir>] - Config directory for the init manager

enable - Enable Cribl LogStream boot-start, args:

[-m <manager>] - Init manager (systemd|initd)

[-u <user>] - User to run Cribl LogStream as

[-c <configDir>] - Config directory for the init manager

diag

Manages diagnostic bundles.

create - Creates diagnostic bundle for Cribl LogStream

list - List existing Cribl LogStream diagnostic bundles

send - Send LogStream 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)

groups

Manages worker groups.

Usage: [sub-command] [options] [args]

Commands:

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            - List worker groups
```

keys

Manages encryption keys.

Usage: [sub-command] [options] [args]

Commands:

```
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
list - List encryption keys
```

mode-searchhead

Configures Cribl LogStream to run on a Splunk Search Head.

o

nc

Listens on a port for traffic, and outputs stats and data. (Netcat-like utility.)

Usage: [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
```

node

Executes a JavaScript file. Displays a command prompt for path/filename input.

>

pipe

Feeds stdin to a pipeline. Examples:

```
cat sample.log | ./cribl pipe -p <pipelineName>
cat sample.log | ./cribl pipe -p <pipelineName> 2>/dev/null
```

scope

Greps your apps by the syscalls. Executes immediately.

splunk-decrypt

Splunk decrypt search command. Executes immediately.

task

Runs a Cribl LogStream task. Requires definitions for the `dir`, `executor`, and `path` properties.

vars

Manages LogStream [Global Variables](#).

Usage: [sub-command] [options] [args]

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 encryption keys, 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

☐ Updated 17 days ago

EXPRESSION REFERENCE

Introduction

As data travels through a Cribl LogStream pipeline, it is operated on by a series of functions. Functions are fundamentally [Javascript](#) code.

Functions that ship with Cribl LogStream 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
<pre>x = 42 newFoo = foo.slice(30)</pre>	<pre>(Math.random() * 42) 3 + 4 'foobar' '42'</pre>

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. They 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'`

Truthy	Falsy
<pre>true 42</pre>	<pre>false null</pre>

-42	undefined
3.14	0
"foo"	NaN
Infinity	' '
-Infinity	'''

Value Expressions

Values 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')`

Considerations and Best Practices for Creating Predictable Expressions

- In a value expression ensure that the source variable is not **null**, **undefined** or **empty**. For example, if you want to have a field called `len` to be assigned the length of a field called `employeeID` but you're not sure if `employeeID` exists, instead of `employeeID.length` you can use a safer shorthand as such: `(employeeID || '').length`.
- If a field does not exist (undefined) and you're doing a comparison with its properties the boolean expression will **always** evaluate to false. For example, if `employeeID` is undefined, then both of these expressions `employeeID.length > 10`, and `employeeID.length < 10` will evaluate to false.
- `==` means equal to, while `===` means equal value and equal type.. For example, `5 == 5` evaluates to **true**, while `5 === "5"` evaluates to **false**.
- 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 can do: `(age >= 18) ? 'adult' : 'minor'`

Expressions Using Fields with Non-Alphanumeric Characters

If there are fields with non-alphanumeric characters, e.g., `@timestamp` or `kubernetes.namespace_name` they can be accessed using `__e['<field-name-here>']`. More details [here](#). On any other place where the field is referenced (e.g., in [Eval](#)'s function field name) a single quoted literal `'<field-name-here>'` should be used.

Wildcard Lists

Wildcards 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 term. They also accept strings that start with 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 whitelists and blacklists.

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 .

 Updated 10 days ago

Cribl Expressions

Native Cribl LogStream 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:

Name	Value Expression
myNewField	<code>C.Mask.sha1(myOtherField)</code>

C.Crypto - Data encryption and Decryption Functions

`C.Crypto.decrypt`

method `Crypto.decrypt(value: 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 where to look for ciphers

@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 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 encryptions fails for any reason, if unspecified the original value is returned

@returns - - if encryption succeeds the encrypted value, otherwise defaultVal if specifier, 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 and returns a string or Buffer depending on 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 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 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 (eg. 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 it's hex representation (lower case). 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;}

An object containing the 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

C.LookupCIDR – CIDR Lookup

(property) Lookup: (file: string, primaryKey?: string, otherFields?: string[]) => InlineLookup

Returns an instance of a CIDR lookup to use inline

C.LookupRegex – Regex Lookup

(property) Lookup: (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 lookup

@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')

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 .

C.Mask – Data Masking Functions

C.Mask.CC

(method) Mask.CC(value: string, unmasked?: number, maskChar?: string): string

Check that 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 unmasked digits, 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 that 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 unmasked digits, 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 that the given value could be a valid credit card number, by computing the string's Luhn'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 that the given value could be a valid IMEI number, by computing the string's Luhn'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 Luhn'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 Luhn'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 Luhn's checksum mod 10 is 0
@param - unmasked - number of unmasked digits, 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, imei etc) By default the mod 10 of the checksum is returned, pass mod = 0 to get actual checksum
@param - value a string whose digits you want to perform the Luhn checksum on
@param - mod return checksum module 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 given value

@param - value compute 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 or the result, or if a string use it's 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 or the result, or if a string use it's length

@param - char pattern which 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 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 values

@param - values - values corresponding to keys

@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 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 if 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.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 if the supplied IPv4 address is in the range of private addresses per [RFC1819](#).

@param - `address` - address to test

C.os – System Functions

`C.confVersion`

Returns Cribl LogStream config version.

`C.os.hostname()`

Returns hostname of system running this Cribl LogStream 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`

e.g., `C.Schema('schema1').validate(myField)` will validate if `myField` object conforms `schema1` .

See [Schema Library](#) 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 compute Shanon entropy of.

@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 compute the hash of

@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, false otherwise.

`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, false otherwise.

`C.Text.relativeEntropy()`

(method) `Text.relativeEntropy(bytes: any, modelName?: string): number`

Computes the relative entropy of the given buffer or string

@param - bytes - value to compute relative entropy of

@param - string modelName - The name of the model to test string with.

@returns - the relative entropy value or -1 in case of an error

C.Time – Time Functions

`C.Time.adjustTZ()`

(method) `Time.adjustTZ(epochTime: number, tzTo: string, tzFrom?: string): number`

Adjust a timestamp from the one timezone to another

@param - epochTime unix epoch time

@param - tzTo timezone to adjust to

@param - tzFrom optional timezone of the timestamp

`C.Time.strftime()`

(method) `Time.strftime(date: number | Date, format: string, utc?: boolean): string`

Format a [Date][1] or number as a time string using [strftime specifier][2] [1]:

https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date [2]:

<https://github.com/d3/d3-time-format#api-reference>

@param - date - Date object or number (seconds since epoc) to format

@param - format - specifier to use to format the date

@param - utc - whether to output the time in UTC rather than 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][2] - if successful a [Date][1] object is returned otherwise null. [1]: https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Date [2]: [https://github.com/d3/d3-time-](https://github.com/d3/d3-time-format#locale_format)

[format#locale_format](https://github.com/d3/d3-time-format#locale_format)

@param - str - string to parse to a timestamp (see strict flag)

@param - format - strftime specifier

@param - utc - whether to interpret times as UTC rather than local time

@param - strict - whether to return null if there are any extra characters after timestamp

@returns - the parsed date or null if the specifier did not match

`C.Time.timestampFinder()`

(method) `Time.timestampFinder(utc?: boolean): AutoTimeParser`

C.vars – Global Variables

See [Global Variables Library](#) for more details.

C.version – Cribl LogStream Version

(property) `version: string`

Cribl LogStream Version

 Updated 4 days ago

KNOWLEDGE

Regex Library

What Is the Regex Library

Cribl LogStream ships with a Regex Library that contains a set of pre-built common regex patterns. The goal of the library is to serve as an easily accessible repository of regular expressions. The library is searchable, and each pattern can be tagged if further organization or categorization is needed. The library can be found under **Knowledge | Regex Library**.

Name	Regex	Tags	Library
> IPv4 address	/ (?<1\d)? (?<[01]? \d\d? 2[0-4] \d 25[0-5]) \.) {3} (?<[01]...	ip, ipv4, address	Cribl
> Email Address	/ [a-zA-Z0-9_+&*~]+ (?< \. [a-zA-Z0-9_+&*~]+)+ @ (?< [a-zA-Z0-9...	email, address	Cribl
> Bitcoin Address	/ [13] [a-km-zA-HJ-NP-Z1-9] {25,34} /gm	bitcoin, btc, address	Cribl
> Mastercard Credit Card	/ 5 [1-5] [0-9] {14} /gm	credit card, mastercard, pii	Cribl
> American Express Credit Card	/ 3 [47] [0-9] {13} /gm	credit card, american express, ...	Cribl
> Discover Credit Card	/ 6 (?< 0115 [0-9] {0-9}) [0-9] {12} /gm	credit card, discover, pii	Cribl
> Visa Credit Card	/ 4 [0-9] {12} [0-9] {3} /gm	credit card, visa, pii	Cribl
> International Bank Account Numbers (IBAN)	/ [a-zA-Z] {2} [0-9] {2} [a-zA-Z0-9] {4} [0-9] {7} ([a-zA-Z0-9] ?...	iban, pii	Cribl
> IPv6 address	/ ((([0-9a-fA-F] {1,4}) {7,7} [0-9a-fA-F] {1,4}) ([0-9a-fA-F] ...	ip, ipv6, address	Cribl
> ISBN-10	/ (?< ISBN (?< -10) ? : ? \s) ? (?< = [0-9K] {10} \$ (?< = (?< [0-9] + [- \s]) ...	isbn, isbn-10	Cribl
> ISBN-13	/ (?< ISBN (?< -13) ? : ? \s) ? (?< = [0-9] {13} \$ (?< = (?< [0-9] + [- \s]) {...	isbn, isbn-13	Cribl
> MAC Address	/ ([0-9a-fA-f] {2} [:-]) {5} ([0-9a-fA-f] {2}) /gm	mac, address	Cribl
> US Phone Numbers	/ ((([0-9] {1})+ [- .])+ ([0-9] {3}) [- .])+ [0-9] {3} [- .])+ [0-9]...	us, phone, number	Cribl
> US Social Security Numbers	/ \d {3} - ? \d {2} - ? \d {4} /gm	us, social, ssn	Cribl
> US State Abbreviated	/ (?< A KLRZ) C AOT D CE FL GA HI I ADLN K SV LA M AD...	us, state	Cribl
> US State	/ Alabama Alaska Arizona Arkansas California Colorado Co...	us, state	Cribl
> US Zip Code	/ \d {5} (?: - \d {4}) ? /gm	us, zip, code	Cribl

How Does It Work

As of this this version, the Library contains 25 patterns shipped by Cribl LogStream. You can use a pattern as-is in a [Function](#), or can modify it as necessary. You can also add new, custom patterns.


```
Rule Foo
Rule Bar
...
Rule FooBar
```

ID	Address	Port	IP Whitelist Regex	TLS	Enabled	Status
<input checked="" type="checkbox"/> myTCPSource	0.0.0.0	7777	/.*	Disabled	<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/> Live

Input Id*

myTCPSource

__inputId== 'tcp:myTCPSource'

Address*

0.0.0.0

Port*

7777

Ip Whitelist Regex

/.*

Enable Header

☐ No

> TLS SETTINGS (SERVER SIDE)

> ADVANCED SETTINGS

> CUSTOM COMMAND PROCESSOR

> EVENT BREAKERS

Event Breaker Rulesets

1

AWS Ruleset Event breaking rules for common AWS data sources

2

Cisco Ruleset Event breaking rules for common Cisco data source

3

Palo Alto Ruleset Event breaking rules for common Palo Alto data source

System Default Rule

Filter Condition: true

Event Breaker: /[\n\r]*(?!\s)/

Timestamp Anchor: /s/

Timestamp Format: Auto:150

Default Timezone: Local

Max Event Bytes: 51200

+ Add Ruleset

Event Breaker Buffer Timeout

10000

> FIELDS (METADATA)

> CONDITIONING PIPELINE

Delete Source

Clone Source

This rule breaks on newlines and uses Manual timestamping **after** the sixth comma, as indicated by this pattern: `^(?:[,]*,){6}`.

System Default Rule

The system default rule sits at the bottom of the ruleset/rule hierarchy, 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

In the **Event Breaker Rulesets** screen (see [above](#)), click **+ Add New** to create a new event breaker ruleset. Click **+ Add Rule** within a ruleset to add a new event breaker.

The screenshot shows the 'Event Breaker Rule' modal window. It has a left sidebar with configuration options and a main area for the rule definition. The sidebar includes fields for 'Rule Name*', 'Filter Condition*' (set to 'true'), 'EVENT BREAKER SETTINGS' (Event Breaker Type* set to 'Regex', Event Breaker* set to '([\n\r]+(?:\s))', Max Event Bytes set to '51200'), 'TIMESTAMP SETTINGS' (Timestamp Anchor* set to '^', Timestamp Format* set to 'Autotimestamp Scan Depth' with a value of '150'), and 'Default Timezone' set to 'Local'. There is also an 'ADD FIELDS TO EVENTS' section with an 'Add Field' button. The main area has tabs for 'In' and 'Out', and a large text area with a dashed border containing the text 'Paste your events here or upload a sample file'. A legend at the bottom left shows 'Timestamp Anchor' (blue), 'Event Breaker' (red), and 'Timestamp' (purple). 'Cancel' and 'OK' buttons are at the bottom right.

Each event breaker includes the following components, which you configure from top to bottom in the above **Event Breaker Rule** modal:

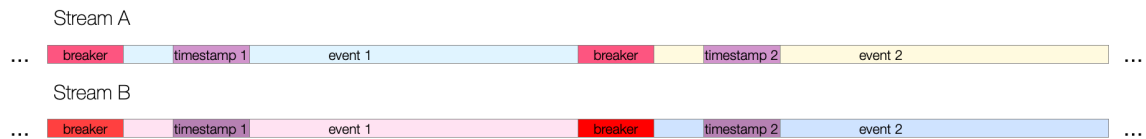
Filter Condition: As a stream of data moves into the engine, a rule's filter expression is applied. If the expression evaluates to `true`, the rule configurations are engaged for the entire duration of that stream. Else, the next rule down the line is evaluated.

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](#).

Timestamping: After events are synthesized out of streams, LogStream 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 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.



Fields: After events have been timestamped, one or more fields can be added. Their values can be fully evaluated using JavaScript expressions.

Event Breaker Types

Several types of Event Breaker can be applied to incoming data streams:

1. **Type Regex** – 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, a positive lookahead regex can be used – e.g., `(?=pattern)` – to keep the content.

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.

Example: Break after a newline or carriage return but only if followed by a timestamp pattern:

Event Breaker: `[\n\r]+(?=\d+-\d+-\d+\s\d+:\d+:\d+)`

Sample Event - Multiline

```
--- input ---
2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, then you can
    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
    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
    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!"
}
```

```

    "_time": 1589920332
}

--- output event 2 ---
{
  "_raw": "2020-05-19 16:32:12 moen3628 ipsum[5213]: Use the mobile TCP feed, 1
  "_time": 1589920332
}

```

2. **Type File Header** – can be used to break files with headers, such as IIS or Bro logs. This type of breaker relies on a header section that lists field names. The header section is typically present at the top of the file, and can be single-line or greater.

After the file has been broken into events, fields will also be extracted, as follows:

- **Header Line:** Regex matching a file header line. For example, `^#`.
- **Field Delimiter:** Field delimiter regex. For example, `\s+`.
- **Field Regex:** Regex with one capturing group, capturing all the fields to be broken by field delimiter. For example, `^#[Ff]ields[:]?s+(.*)`
- **Null Values:** Representation of a null value. Null fields are not added to events.
- **Clean Fields:** Whether to clean up field names by replacing non `[a-zA-Z0-9]` characters with `_`.

Example: Using the values above, let's see how this sample file breaks up:

Sample Event - File Header

```

--- input ---
#fields ts      uid      id.orig_h      id.orig_p      id.resp_h      id.resp_p
#types time     string  addr    port    addr    port    enum
1331904608.080000 -      192.168.204.59 137      192.168.204.255 137
1331904609.190000 -      192.168.202.83 48516    192.168.207.4  53

--- output event 1 ---
{
  "_raw": "1331904608.080000 -      192.168.204.59 137      192.168.204.255 137",
  "ts": "1331904608.080000",
  "id_orig_h": "192.168.204.59",
  "id_orig_p": "137",
  "id_resp_h": "192.168.204.255",
  "id_resp_p": "137",
  "proto": "udp",
  "_time": 1331904608.08
}

--- output event 2 ---
{
  "_raw": "1331904609.190000 -      192.168.202.83 48516    192.168.207.4  53",
  "ts": "1331904609.190000",
  "id_orig_h": "192.168.202.83",
  "id_orig_p": "48516",
  "id_resp_h": "192.168.207.4",
  "id_resp_p": "53",

```

```

    "proto": "udp",
    "_time": 1331904609.19
}

```

3. **Type JSON Array** – can be used to extract events from an array in a JSON document (e.g., an Amazon CloudTrail file).

- **Array Field:** 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` .
- **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.

Example: Using the values above, let's see how this sample file breaks up:

Sample Event - JSON Document (Array)

```

--- input ---
{"Records":[{"eventVersion":"1.05","eventTime":"2020-04-08T01:35:55Z","eventSource":"ec2.amazonaws.com","eventName":"StartInstances","requestParameters":{"instanceCount":"1","instanceProfile":"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile"},"responseElements":{"instances":[{"id":"i-12345678","imageId":"ami-12345678","instanceProfile":"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile","keyName":"","monitoring":{"state":"disabled"},"placement":{"availabilityZone":"us-east-1a"},"platform":"","platformOptions":{"architecture":"x86_64"},"securityGroups":["sg-12345678"],"subnetId":"subnet-12345678","tags":[{"key":"Name","value":"my-instance"}]}]}]}]}

--- output event 1 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:55Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"StartInstances\",\"requestParameters\":{\"instanceCount\":\"1\",\"instanceProfile\":\"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile\"},\"responseElements\":{\"instances\":[{\"id\":\"i-12345678\",\"imageId\":\"ami-12345678\",\"instanceProfile\":\"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile\",\"keyName\":\"\",\"monitoring\":{\"state\":\"disabled\"},\"placement\":{\"availabilityZone\":\"us-east-1a\"},\"platform\":\"\",\"platformOptions\":{\"architecture\":\"x86_64\"},\"securityGroups\":[\"sg-12345678\"],\"subnetId\":\"subnet-12345678\",\"tags\":[{\"key\":\"Name\",\"value\":\"my-instance\"}]}]}]}\",
  "_time": 1586309755,
  "cribl_breaker": "j-array"
}

--- output event 2 ---
{
  "_raw": "{\"eventVersion\":\"1.05\",\"eventTime\":\"2020-04-08T01:35:56Z\",\"eventSource\":\"ec2.amazonaws.com\",\"eventName\":\"StartInstances\",\"requestParameters\":{\"instanceCount\":\"1\",\"instanceProfile\":\"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile\"},\"responseElements\":{\"instances\":[{\"id\":\"i-12345679\",\"imageId\":\"ami-12345679\",\"instanceProfile\":\"arn:aws:iam::123456789012:instance-profile/EC2-Instance-Profile\",\"keyName\":\"\",\"monitoring\":{\"state\":\"disabled\"},\"placement\":{\"availabilityZone\":\"us-east-1a\"},\"platform\":\"\",\"platformOptions\":{\"architecture\":\"x86_64\"},\"securityGroups\":[\"sg-12345679\"],\"subnetId\":\"subnet-12345679\",\"tags\":[{\"key\":\"Name\",\"value\":\"my-instance-2\"}]}]}]}\",
  "_time": 1586309756,
  "cribl_breaker": "j-array"
}

```

4. **Type JSON New Line Delimited** – can be used to break and extract fields in newline-delimited JSON streams.

Example: Using default values, let's see how this sample stream breaks up:

Sample Event - Newline Delimited JSON

```

--- input ---
{"time":"2020-05-25T18:00:54.201Z","cid":"w1","channel":"clustercomm","level":"info","message":"Cluster comm message"}
{"time":"2020-05-25T18:00:54.246Z","cid":"w0","channel":"clustercomm","level":"info","message":"Cluster comm message"}

--- output event 1 ---
{
  "_raw": "{\"time\":\"2020-05-25T18:00:54.201Z\",\"cid\":\"w1\",\"channel\":\"clustercomm\",\"level\":\"info\",\"message\":\"Cluster comm message\"}\",
  "time": "2020-05-25T18:00:54.201Z",
  "cid": "w1",
  "channel": "clustercomm",
  "level": "info",
  "message": "Cluster comm message"
}

```

```

    "message": "metric sender",
    "total": 720,
    "dropped": 0,
    "_time": 1590429654.201,
  }

--- output event 21 ---
{
  "_raw": "{\\"time\\":\\"2020-05-25T18:00:54.246Z\\",\\"cid\\":\\"w0\\",\\"channel\\":\\"'
  "time": "2020-05-25T18:00:54.246Z",
  "cid": "w0",
  "channel": "clustercomm",
  "level": "info",
  "message": "metric sender",
  "total": 720,
  "dropped": 0,
  "_time": 1590429654.246,
}

```

Cribl versus Custom Rulesets

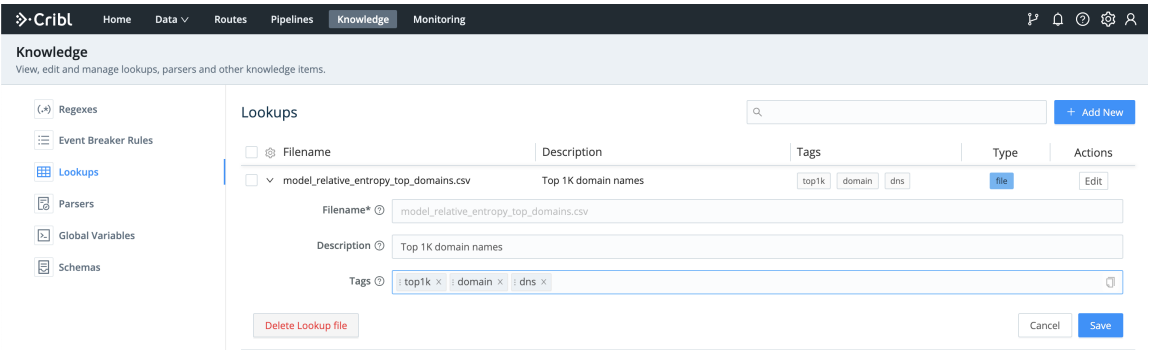
Event Breaker rulesets shipped by Cribl will be listed under the **Cribl** tag, while user-built rulesets will be found under **Custom**. Over time, Cribl will ship more patterns, so this distinction allows for both sets to grow independently. In the case of an ID/Name conflict, the Custom pattern takes priority in listings and search.

 Updated 3 days ago

Lookups Library

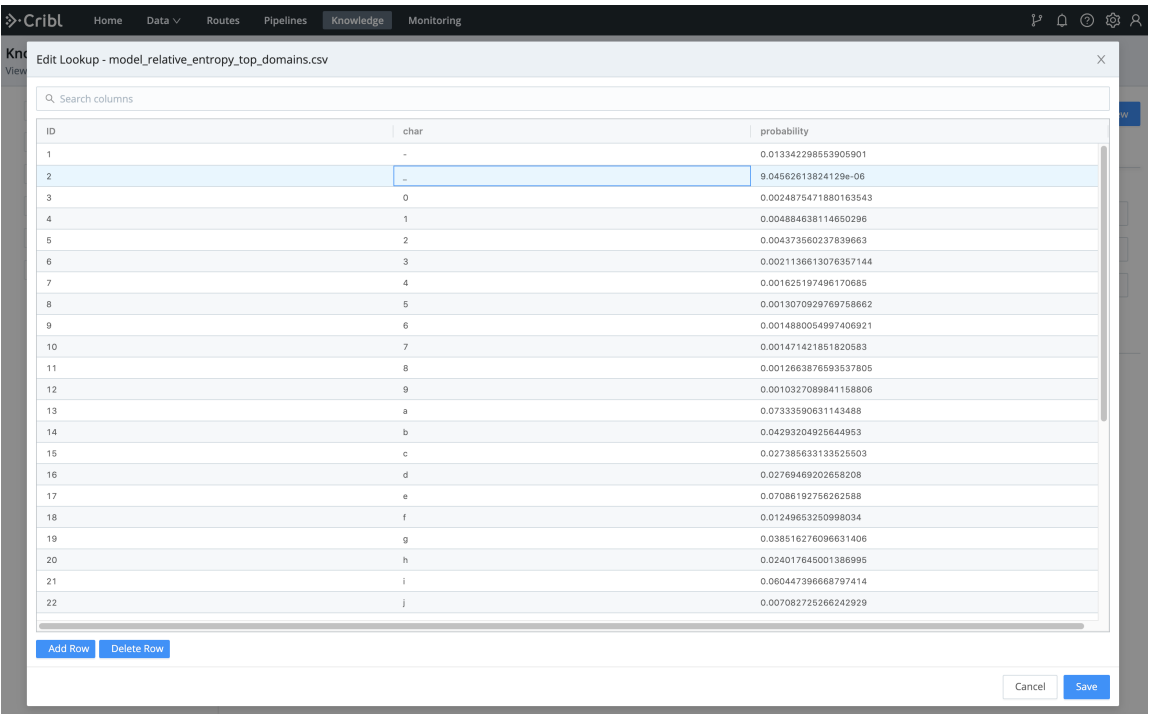
What Are Lookups

Lookups are data tables that can be used in Cribl LogStream to enrich events as they're processed by the [Lookup Function](#). The Lookups library can be found under **Knowledge > Lookups**, and its goal is to provide a management interface for all lookups. The library is searchable, and each lookup can be tagged as necessary.



How Does It Work

The management interface allows for lookups to be added, deleted and edited. All files handled by the interface are stored in `$CRIBL_HOME/data/lookups`.



 Updated 3 days ago

Parsers Library

What Are Parsers

Parsers are definitions and configurations for the [Parser Function](#). The library can be found under **Knowledge > Parsers** and its goal is to provide an interface for creating and editing Parsers. The library is searchable and each parsers can be tagged as necessary.

Parsers			
(.*) Regexes			
Event Breakers			
Lookups			
Parsers			
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 can be used to **extract** or **reserialize** events. See [Parser Function](#) page for examples.

Supported Parser Types:

- CSV - parse and reserialize comma separated values.
- CLF - parse and reserialize events in Common Log Format.
- ELFF - parse and reserialize events in Extended Log File Format

Creating a Parser

To create a parser, follow these steps:

- Go to **Knowledge > Parsers** and click **Add New**.
- Enter a name/ID.
- Select a parser type (see supported types above).
- Enter the list of fields expected to be extracted, in order.
 - Click on the maximize icon (far right) if you'd like to work with sample data and iterate on results.

Updated 3 days ago

Schema Library

What Are Schemas

Schemas are JSON definitions that are used to validate of JSON events. They're based on the popular [JSON Schema standard](#). and all schemas matching draft version 2019-09 are supported. The library can be found under **Knowledge | Schemas** and its goal is to provide an interface for creating, editing and maintaining Schemas.

Schema validation is done using `C.Schema('<schema name>').validate(<object field>)` function. This function can be called anywhere where JS expressions are supported in the product.

Typical use cases for Schema validation:

- Making a decision before sending an event down to a destination.
- Making a decision before accepting an event. I.e., drop event if invalid.
- Making a decision to route an event given the result of validation.

Example

To add this example JSON Schema go to **Knowledge | Schemas** and click **Add New**. Supply:

- Schema Id: schema1
- Description: Enter your own description here
- Schema: Paste the following schema

JSON Schema - Sample

```
{
  "$id": "https://example.com/person.schema.json",
  "$schema": "http://json-schema.org/draft-07/schema#",
  "title": "Person",
  "type": "object",
  "required": ["firstName", "lastName", "age"],
  "properties": {
    "firstName": {
      "type": "string",
      "description": "The person's first name."
    },
    "lastName": {
      "type": "string",
      "description": "The person's last name."
    },
    "age": {
```

```

    "description": "Age in years which must be equal to or greater than zero.",
    "type": "integer",
    "minimum": 0,
    "maximum": 42
  }
}
}

```

Assume that events look like this:

Events

```

{"employee":{"firstName": "John", "lastName": "Doe", "age": 21}}
{"employee":{"firstName": "John", "lastName": "Doe", "age": 43}}
{"employee":{"firstName": "John", "lastName": "Doe"}}

```

To validate whether the `employee` field is valid per `schema1` we can use the following:

```
C.Schema('schema1').validate(employee)
```

Results:

- First event is **valid**.
- Second event is **not valid** because `age` is greater than the maximum of `42`.
- Third event is **not valid** because `age` is missing.

#FunctionFilter▼

1Evaltrue

Filter ⓘ

true

Description ⓘ

Enter a description

Final ⓘ

No

Evaluate Fields ⓘ

Name ⓘValue Expression ⓘ

isValidC.Schema('schema1').validate(employee)

+ Add Field

Keep Fields ⓘ

Enter field names

Remove Fields ⓘ

Enter field names

Show All ⓘ

On ...

EVENTTABLE

#Event

1#_time: 1578681447.911
2020-01-09 15:24:07.911
-05:00
 @cribl_breaker: Break on newlines
 @cribl_ptipe: myPipeline
 @employee:
 # age: 21
 # firstName: John
 # lastName: Doe
 @isValid: true

2#_time: 1578681447.911
2020-01-09 15:24:07.911
-05:00
 @cribl_breaker: Break on newlines
 @cribl_ptipe: myPipeline
 @employee:
 # age: 43
 # firstName: John
 # lastName: Doe
 @isValid: false

3#_time: 1578681447.911
2020-01-09 15:24:07.911
-05:00
 @cribl_breaker: Break on newlines
 @cribl_ptipe: myPipeline
 @employee:
 # firstName: John
 # lastName: Doe
 @isValid: false

Updated 10 days ago

Global Variables Library

What Are Global Variables

Global Variables are reusable JS expressions that can be accessed in [Functions](#) in any [pipeline](#). The library can be found under **Knowledge | Global Variables**.

Typical use-cases for Global Variables include the following:

- 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 where JS expressions are supported in the product. 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 LogStream 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 LogStream 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 fields `size` in human readable format.

- Global Variable Name: `convertBytes` <-- ships with Cribl LogStream 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)))]}``
- Global Variable Argument: `bytes`
- **Sample Eval Function:** `storage = C.vars.convertBytes(size)`

Note the use of `bytes` here as an argument.

 Updated 11 days ago

USE CASES

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 leave it empty and apply it 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:



The screenshot shows the configuration for an **Eval** function in a Splunk pipeline. The top bar indicates the function is named **Eval** and is currently **On**. The filter is set to `sourcetype=='access_combined'`. The description field contains the text "Add 'dc' index-time field to events". The **Final** toggle is set to **No**. Under **Evaluate Fields**, there is a button labeled **+ Add Field**. The **Keep Fields** and **Remove Fields** sections each have a text input field labeled "Enter field names".

- Next, let's click on **+ Add Field**, add our `dc` field, and click **Save**.

3 Eval `sourcetype=='access_combined'` On ...

Filter ?

`sourcetype=='access_combined'`

Description ?

Add 'dc' index-time field to events

Final ? ☐ No

Evaluate Fields ?

Name ?	Value Expression ?
dc	'nyc-42'

+ Add Field

Keep Fields ?

Enter field names

Remove Fields ?

Enter field names

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:

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'

+ Add Field

Keep Fields ?

Enter field names

Remove Fields ?

Enter field names

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

Removing fields can be done by either listing or wildcarding of 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'

+ Add Field

Keep Fields [?](#)

Enter field names

Remove Fields [?](#)

date_*

Cancel Save ...

To confirm, verify that this search: `sourcetype="access_combined" date_minute=*` will soon stop returning results. Enjoy a more efficient Splunk!

Updated 3 days ago

Ingest-time Lookups

Enriching Data in Motion

To enrich events with new fields from external sources – say, `.csv` files – we use the 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, when you have a highly dynamic infrastructure, and you need to resolve a resource (e.g., a container) name to its address, you can't afford to do it at search time/runtime, as the resource and its records might no longer exist. External (non-`.csv`) lookups are coming soon.

Working with Lookups – Example 1

Let's assume we have the following lookup file. Given the field `conn_state` in an event, we would like to add a corresponding ingestion-time field called `action`.

`bro_conn_state.csv`

```
action,"conn_state","conn_state_meaning"
dropped,S0,"Connection attempt seen, no reply."
allowed,S1,"Connection established, not terminated."
allowed,SF,"Normal establishment and termination."
blocked,REJ,"Connection attempt rejected."
allowed,S2,"Connection established and close attempt by originator seen (but not allowed)."
allowed,S3,"Connection established and close attempt by responder seen (but not allowed)."
allowed,RST0,"Connection established, originator aborted (sent a RST)."
allowed,RSTR,"Established, responder aborted."
dropped,RST0S0,"Originator sent a SYN followed by a RST, we never saw a SYN-ACK."
dropped,RSTRH,"Responder sent a SYN ACK followed by a RST, we never saw a SYN ACK."
dropped,SH,"Originator sent a SYN followed by a FIN, we never saw a SYN ACK from responder."
dropped,SHR,"Responder sent a SYN ACK followed by a FIN, we never saw a SYN ACK from originator."
allowed,OTH,"No SYN seen, just midstream traffic (a 'partial connection' that was not terminated)."

```

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`

3
Lookup
sourcetype=="bro"
On

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	X

+ Add field(s)

Output field(s) ?

	Output Field Name from Lookup ?	Lookup Field Name in Event ?	Default Value ?	
	action	action	Enter default value	X

+ Add field(s)

To confirm, verify that this search returns expected results: `sourcetype="bro"`
`action::allowed` . Change the `action` value as necessary.

Working with Lookups – Example 2

Let's assume we have the following lookup file, and given **both** the fields `impact` and `priority` in an event, we would like to add a corresponding ingestion-time field called `severity` .

cisco_sourcefire_severity.csv

```

impact,priority,severity
1,high,critical
2,high,critical
3,high,high
4,high,high
0,high,high
"*,high,high
.....
"*,medium,medium
1,low,medium

```

```

2, low, medium
3, low, low
4, low, low
0, low, low
"*, low, low
1, none, low
2, none, low
3, none, informational
4, none, informational
0, none, informational
"*, none, informational

```

First, make sure you have a route and pipeline configured to match desired events.

Next, let's add a Lookup function to the pipeline, with these settings:

- **Lookup file path:**
\$SPLUNK_HOME/etc/apps/Splunk_TA_sourcefire/lookups/cisco_sourcefire_severity.csv
(Note that Environment variables are allowed in the path.)
- **Lookup Field Name(s) in Event** set to `impact` and `priority`
- **Corresponding Field Name(s) in Lookup** set to `impact` and `priority`
- **Output Field Name from Lookup** set to `severity`
- **Lookup Field Name in Event** set to `severity`

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	X
	priority	priority	X

+ Add field(s)

Output field(s)

	Output Field Name from Lookup	Lookup Field Name in Event	Default Value	
	severity	severity	Enter default value	X

+ Add field(s)

To confirm, verify that this search returns expected results:

`sourcetype="cisco:sourcefire" severity::medium` . Change the `severity` value as necessary.



Updated 3 days ago

Sampling

Sampling at ingest-Time

Let's say that you wanted troubleshoot with, and analyze, **highly verbose/voluminous** data – for example, CDN logs, ELB Access Log, 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, but you can also do all the troubleshooting necessary.

See the example below, or 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' at 5:1 (and all other events 1:1). This should lower the volume of all verbose successes (200s), but still bring in **all** potentially erroneous events (400s, 500s, 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 LogStream, and can be used anywhere in a pipeline.

The screenshot shows the 'Regex Extract' configuration window in Cribl LogStream. At the top, it indicates step 3 of the configuration, the function name 'Regex Extract', and the filter `sourcetype=='access_combined'`. A toggle switch on the right is set to 'On'. The 'Filter' section contains the same filter expression. The 'Description' section has the text 'Extract status from access logs'. The 'Final' section has a toggle switch set to 'No'. The 'Regex*' section contains the pattern `/ \"\s(?<__status>\d+)`. Below this is an 'Additional Regex' section with a '+ Add Regex' button. The 'Source Field' section has a dropdown menu with `_raw` selected.

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 ?
<div> <div>⋮</div> <div>__status == 200</div> <div>⌵</div> </div>	<div>5</div> <div>×</div>

+ Add Rule

To confirm that sampling works, compare the event count of all 200 s before and after. In addition, each time an event goes thru the Sampling function an index-time field is added to it `samplerate::<rate>` . You can use that to in your statistical functions as necessary.

 Updated 3 days ago

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), and 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+)/`

3

Regex Extract

sourcetype=='access_combined' || sourcetype=='aw... On ...

Filter ?

sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'

Description ?

Extract status from access logs

Final ? No

Regex* ?

/ HTTP\/\d\.\d"\"s(?<__status>\d+)

Additional Regex

+ Add Regex

Source Field ?

_raw

3. Add a Sampling function to sample 5:1 when __status==200 .

4. Save.

4

Sampling

sourcetype=='access_combined' || sourcetype=='aw... On ...

Filter ?

sourcetype=='access_combined' || sourcetype=='aws:s3:accesslogs'

Description ?

Check for status 200 and sample 5:1

Final ? No

Sampling Rules ?

Filter ?	Sampling Rate ?
__status == 200	5

+ Add Rule

Note About Sampling

Each time an event goes thru 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 may need a different `__status` extraction regex:

Amazon Cloudfront Access Logs	sourcetype=='aws:cloudfront:accesslogs'
Amazon ELB Access Logs	sourcetype=='aws:elb:accesslogs'

 Updated 3 days ago

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 LogStream 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

The [VPC Flow Logs](#) feature enables you to capture information about the IP traffic going to and from network interfaces in your VPC. Flow log data can be published to Amazon CloudWatch Logs and Amazon S3.

Typical VPC Flow Logs look like this:

Flow Log Records for Accepted and Rejected Traffic

```
2 123456789010 eni-abc123de 172.31.16.139 172.31.16.21 20641 22 6 20 4249 1418!  
2 123456789010 eni-abc123de 172.31.9.69 172.31.9.12 49761 3389 6 20 4249 14185:
```

Let's use a **very simple** filter condition and only look for `ACCEPT` s.

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

- Add a Sampling function to sample 5:1 when __action=="ACCEPT" .
- 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

Note About Sampling

Each time an event goes thru 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 may need a different __action extraction regex:

Cisco ASA Logs	sourcetype=='cisco:asa' Related sourcetypes to consider sampling: sourcetype=='cisco:fwsn' sourcetype=='cisco:pix'
----------------	--

 Updated 3 days ago

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 Masking function accepts multiple replacement rules and 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:

7

Mask

source=='pii_naivecc'

On

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)

+ Add Rule

Apply to Fields ?

:_raw

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`

i Replace Expression Evaluation

Replace Expression 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"```

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.

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:** `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=XXXXXXXXXXXXXXXX`

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.

 Updated 3 days ago

Regex Filtering

Regex Filtering of Data in Motion

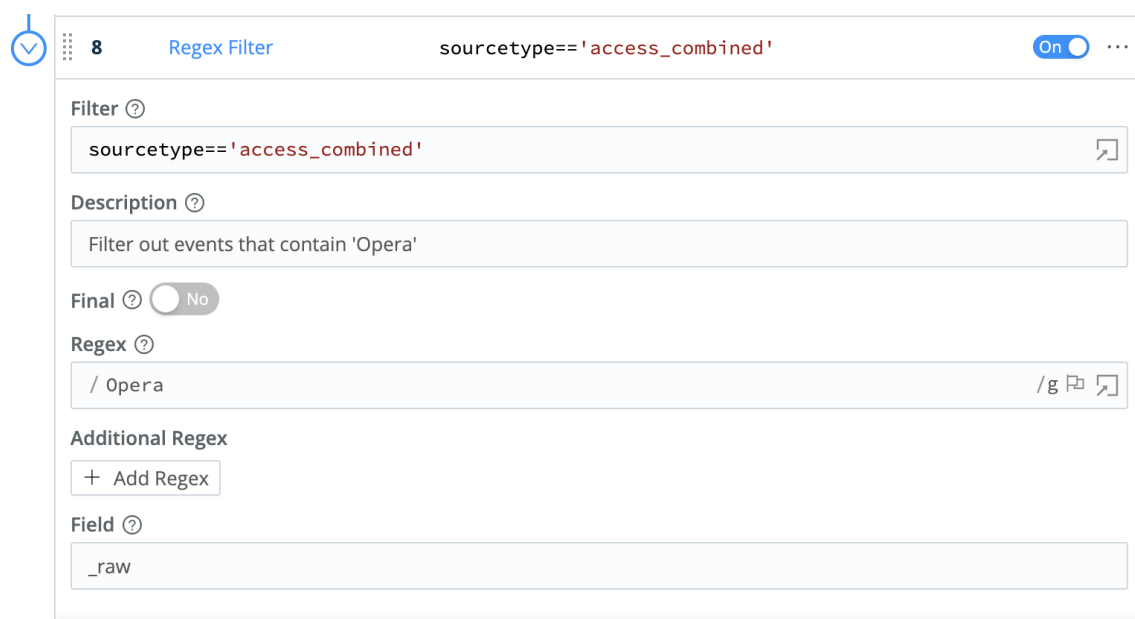
To filter events in real-time we use the out-of-the-box **Regex Filter** function. This is similar to nullqueueing with TRANSFORMS in Splunk, but the matching condition is way more flexible.

Regex Filtering Example

Let's see how we can filter out any `sourcetype=='access_combined'` events that contain the pattern `Opera` in `_raw` :

First, make sure you have a route and pipeline configured to match desired events.

Next, let's add a **Regex Filter** function to it:



The screenshot shows the configuration for a 'Regex Filter' function in Splunk. The top bar indicates the function is active (On) and shows the filter expression: `sourcetype=='access_combined'`. The configuration fields are as follows:

- Filter**: `sourcetype=='access_combined'`
- Description**: Filter out events that contain 'Opera'
- Final**: No (toggle switch)
- Regex**: `/ Opera`
- Additional Regex**: + Add Regex
- Field**: `_raw`

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:

8

Regex Filter

`sourcetype=='access_combined' && host.endsWith('...`

On

Filter ?

`sourcetype=='access_combined' && host.endsWith('dnto.ca')`

Description ?

Filter out events that contain 'Opera'

Final ?

No

Regex ?

`/ Opera`

/g

Additional Regex

+ Add Regex

Field ?

`_raw`

This is a very flexible method for filtering incoming events in real time, on virtually any arbitrary conditions.

 Updated 3 days ago

Encrypting Sensitive Data

Encryption at Ingest-Time

With Cribl LogStream 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.

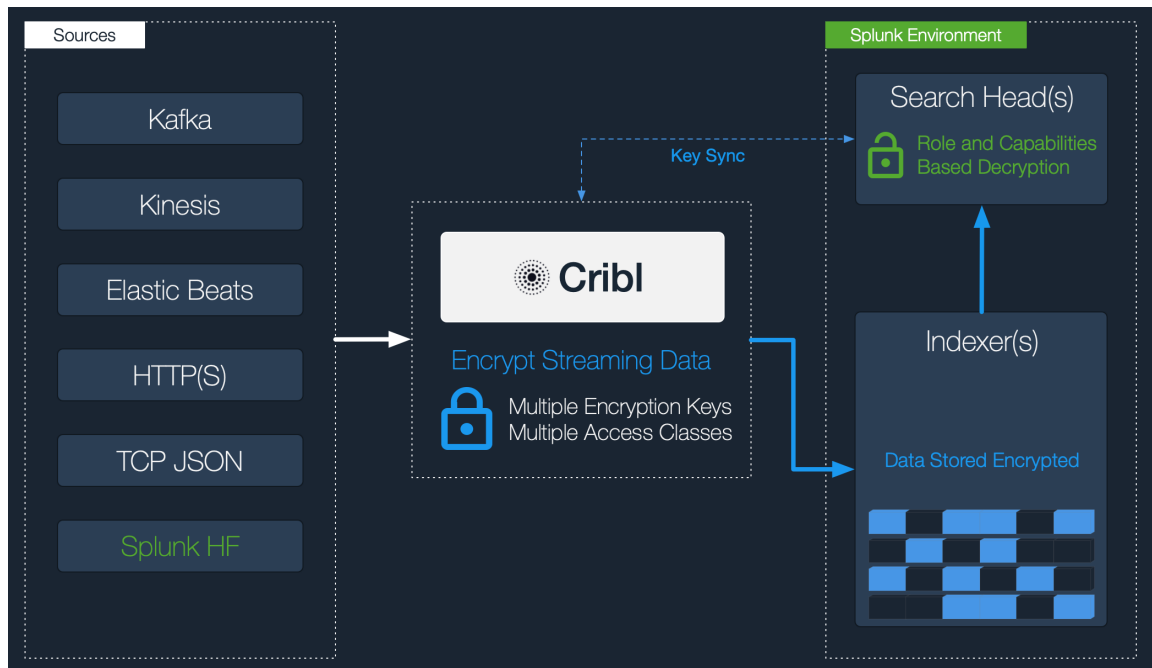
Keys and Key Classes

Symmetric encryption [keys](#) can be configured through the CLI or 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) with access to data with encrypted patterns can be associated with key classes, for even more granular, pattern-level, compartmentalized access.

Encrypting in Cribl LogStream and Decrypting in Splunk

1. Define one or more [Keys](#) and [Key Classes](#) on Cribl LogStream.
2. Sync `auth` with the decryption side (Splunk Search Head)
3. Apply the [Mask](#) function with [C.Crypto.encrypt\(\)](#) to patterns of interest.
4. Decrypt on Splunk Search Head, using Role Based Access Control on the `decrypt` command.



Example

Encryption Side

- Generate one or more keys via the CLI, as such:

```
$CRIBL_HOME/bin/cribld keys add -c 1 -i
```

...

```
$CRIBL_HOME/bin/cribld keys add -c <N> -i
```

Add `-e <epoch>` if you'd like to set expiration for your keys.

Or via the UI, in Settings > Encryption Keys:

Manage Encryption Keys
+ Add New

To protect against accidental changes, key parameters can *only* be modified through configuration files.

Get Key Bundle

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

- Sync `auth/(cribl.secret|keys.json)`. To decrypt data, the `decrypt` command will need access to these keys. The `cribl.secret` and `keys.json` files in

`$CRIBL_HOME/local/cribl/auth` should be synced/copied over to the Search Head (decryption side).

Decryption Side

- Install Cribl App for Splunk on your Search Head. It 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 >

Selected capabilities

< clear 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

← 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

Usage

Before Encryption: sample un-encrypted events. Notice values of `fieldA` and `fieldB`

index=main source=wifi.log fieldA=* OR fieldB=* OR fieldC=*			All time	Q
✓ 4 events (before 1/5/19 1:11:25.000 PM) No Event Sampling			Job	Smart Mode
Events (4) Patterns Statistics Visualization				
Format Timeline Zoom Out Zoom to Selection Deselect 1 hour per column				
Show Fields List Format 20 Per Page				
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		

Encrypting `fieldA` values with key class 1 and `fieldB` with key class 2

Page 321 of 326

2 Mask true

On X

Filter ?

true

Description ?

Final ?

Capture field values

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 X

After Encryption: again, notice values of fieldA and fieldB

index=main source=wifi.log fieldA=* OR fieldB=* OR fieldC=*

All time Q

4 events (before 1/5/19 1:19:21.000 PM) No Event Sampling

Job + - + - Smart Mode

Events (4) Patterns Statistics Visualization

Format Timeline - Zoom Out + Zoom to Selection X Deselect 1 hour per column

Show Fields List Format 20 Per Page

i	Time	Event
>	12/20/18 12:44:02.573 PM	Sat Dec 20 12:44:02.573 <kernel> fieldA=#2::1B76682//ZM0NrgyhQnJ8g# fieldB=#3::1Nb0p8LA8EH7nYwFsfBg# 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::1B76682//ZM0NrgyhQnJ8g# fieldB=#3::1Nb0p8LA8EH7nYwFsfBg# 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::1B76682//ZM0NrgyhQnJ8g# fieldB=#3::1Nb0p8LA8EH7nYwFsfBg# 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::1B76682//ZM0NrgyhQnJ8g# fieldB=#3::1Nb0p8LA8EH7nYwFsfBg# fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267

Decrypting fieldB but not fieldA . Logged in user has been assigned capability cribl_keyclass_2

index=main source=wifi.log fieldA=* OR fieldB=* OR fieldC=* | decrypt

All time Q

4 events (before 1/5/19 1:23:16.000 PM) No Event Sampling

Job + - + - Smart Mode

Events (4) Patterns Statistics Visualization

Format Timeline - Zoom Out + Zoom to Selection X Deselect 1 hour per column

Show Fields List Format 20 Per Page

i	Time	Event
>	12/20/18 12:44:02.573 PM	fieldA=#2::1B76682//ZM0NrgyhQnJ8g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267
>	12/20/18 9:15:03.467 AM	fieldA=#2::1B76682//ZM0NrgyhQnJ8g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267
>	12/20/18 9:15:02.590 AM	fieldA=#2::1B76682//ZM0NrgyhQnJ8g# 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::1B76682//ZM0NrgyhQnJ8g# fieldB=SECRET fieldC=MODERATELYSECRET IOCTL (from pid 183) not recognized: 5 out of 267

Updated 3 days ago

KNOWN ISSUES

Known Issues

Can't switch from Worker to Master Mode (2020-06-11)

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 progress.

Workaround: To initialize `git`, switch first from Worker to Single mode, and then from Single to Master mode.

Login page blocks (2020-05-19)

Problem: Entering valid credentials on the login page (e.g., `http://localhost:9000/login`) yields only a spinner.

Fix: In progress.

Workaround: Trim `/login` from the URL.

Deleting resources in `default/` (2020-02-22)

Problem: In a Distributed deployment, deleting resources in `default/` causes them to reappear on restart.

Workaround/Fix: In progress.

In-product upgrade issue on v2.0 (2019-10-22)

Problem: Using in-product upgrade feature in v1.7 (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](#).

In-product upgrade issue on v1.7 (2019-08-27)

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](#).

S3 stagePath issue on upgrade (2019-03-21)

Problem: When upgrading from v1.2 with a S3 output configured `stagePath` was allowed to be undefined. In v1.4+ it is a required field and may causing schema violations on older configs when upgrading.

Workaround/Fix: Re-configure the output with a valid `stagePath` filesystem path.



Updated 3 days ago

THIRD PARTY SOFTWARE

Credits

Various components in Cribl LogStream are built and enhanced with software under free or open source licenses. We thank the contributors of those projects!

@azure-storage-blob-10.3.0

ag-grid-community-19.1.2

ag-grid-react-19.1.2

ajv-6.9.2

ajv-errors-1.0.1

antd-3.13.0

as-table-1.0.36

avsc-5.4.9

aws-sdk-2.323.0

cidr-matcher-1.0.5

classnames-2.2.6

color-hash-1.0.3

d3-time-format-2.1.3

date-fns-1.29.0

diff-3.5.0

diff2html-2.11.3

echarts-4.3.0

echarts-4.6.0

escodegen-1.11.1

esprima-4.0.1

express-4.16.3

fast-bitset-1.3.2

file-saver-1.3.8

http-proxy-agent-3.0.0

https-proxy-agent-4.0.0

jwt-simple-0.5.6

kafkajs-1.11.0

kafkajs-1.4.5

kafkajs-snappy-1.1.0

ldapts-1.10.0

limiter-1.1.4

lodash-4.17.15

lz4js-0.2.0
maxmind-3.1.2
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-16.7.0
react-dom-16.7.0
react-grid-layout-0.16.6
react-router-dom-4.3.1
react-sortable-hoc-0.8.3
react-split-pane-0.1.82
regexp-2.0.0
requirejs-2.3.6
resize-observer-polyfill-1.5.0
rxjs-6.2.2
saxen-8.1.0
simple-git-1.126.0
snappyjs-0.6.0
snmp-native-1.2.0
streamcount-1.0.1
tar-stream-1.6.1
url-0.11.0
winston-3.0.0
xmlbuilder-10.0.0
yaml-1.3.2

 Updated 3 months ago