

Cribl Lake Documentation Manual

Version: v4.13

Generated: 2025-08-27 07:54:52

1. About Cribl Lake	3
2. Lake Datasets	6
2.1. Manage Lake Datasets	8
3. Search Cribl Lake	12
4. Lakehouses in Cribl Lake	16
4.1. Streamline VPC Flow Logs Analysis with a Lakehouse	23
4.2. Streamline CloudTrail Logs Analysis with a Lakehouse	28
4.3. Streamline CloudFront Logs Analysis with a Lakehouse	33
5. Cribl Lake Permissions	38
6. Cribl Lake Collector	39
7. Cribl Lake Destination	42
8. Cribl Lake Direct Access	45
9. Integrating Cribl Lake with Cribl Edge	49
10. Known Issues	51

1. ABOUT CRIBL LAKE

Meet Cribl Lake, a data lake solution for long-term, full-fidelity storage of IT and security data.

Cribl Lake receives data from various sources via [Cribl Stream](#), and stores that data – enabling you to later query the data in [Cribl Search](#), or replay it through Stream as needed.

Besides data you explicitly send to the Lake, Cribl Lake also receives and retains internal logs and metrics from the Leader of your Cribl.Cloud Organization.

What Can You Do with Cribl Lake?

Among the features that Cribl Lake provides, you can:

- Create a data lake in a few clicks and a few minutes.
- Store any type of IT and security data: raw, structured, or unstructured.
- Set and manage [retention](#) and access-control policies through a graphical UI, reducing dependencies among multiple teams.
- Use Cribl Stream [Pipelines](#) and Functions to control the mix, volume, and structure of data sent to your lake.
- Search your Lake data in place, using Cribl Search, to minimize ingress charges for analysis.
- Assign Lake Datasets to [Lakehouses](#) for dramatically faster search responses and more-predictable billing.

Requirements for Cribl Lake

Cribl Lake is available only in [Cribl.Cloud](#).

Cribl Lake can connect, via a [Collector](#) or [Destination](#), with both Cribl-managed [Cloud](#) and customer-managed [hybrid](#) Stream Worker Groups. Hybrid Worker Groups must be running version 4.8 or higher.

Data Storage in Cribl Lake

Cribl Lake organizes your data into [Datasets](#).

Filter or search	Filter destinations	Filter collectors	Retention period	Add Dataset
☐ ☐ ID & Description	Retention Period	Accelerated Fields	Integrated With	Actions
☐ default_events Think events from Kubernetes or ...	30 days		None	Search
☐ default_logs Default dataset for capturing logs...	30 days		1 Destination & 1 Collector	Search
☐ default_metrics Default dataset for capturing met...	15 days		None	Search
☐ default_spans Default dataset for capturing dist...	10 days		None	Search
☐ cribl_logs Internal logs from Cribl deployme...	30 days		None	Search
☐ cribl_metrics Internal metrics from Cribl deploy...	30 days		None	Search
☐ sampl dataset My sample dataset	365 days	hostname	1 Destination	Search

Built-in and custom Datasets in Cribl Lake.

Your data is stored in open, non-proprietary formats, which avoids vendor lock-in. At any time, you can replay the stored data and send it to a [Destination](#) of your choice.

Cribl Lake takes care of the storage format schema and organizes the data for you. This way, you don't have to think about your partitioning configuration.

The data is stored as gzip-compressed JSON, or Parquet, files.

Storage Limits in Cribl Lake

With a [Cribl.Cloud](#) free plan, you can store up to 50 GB of data.

Data in the `cribl_metrics` and `cribl_logs` Datasets is stored for 30 days free of charge.

Send Data to Cribl Lake

You can archive data to Cribl Lake in several ways:

- To send data from any Cribl Stream Source to Cribl Lake, use the Stream [Cribl Lake Destination](#).
- To [send the results](#) of a Cribl Search query to Cribl Lake, use the [export](#) operator.
- To archive Splunk Cloud DDSS (Dynamic Data Self Storage) data to Cribl Lake, use the Lake [Direct Access](#) option.

Replay Data from Cribl Lake

Cribl Lake adds a dedicated [Collector Source](#) to Cribl Stream, which lets you replay data stored in the lake and send it via Routes or QuickConnect.

To replay data stored in Cribl Lake and send it to any [Destination](#) via Routes or QuickConnect, use the [Cribl Lake Collector](#) available in Cribl Stream.

Search Cribl Lake

[Cribl Lake Datasets](#) work as [Cribl Search Datasets](#) out-of-the-box.

This means you do not need to create any additional [Dataset Providers](#) or [Datasets](#) in Cribl Search. You can instantly [start searching](#) the data flowing to your Cribl Lake.



See [Searching Cribl Lake](#) for example searches you can run over your data stored in Cribl Lake.

Monitor Cribl Lake Usage

You can monitor your Cribl Lake data usage on the Cribl.Cloud portal's [FinOps Center](#) page.

Details of credits consumed by Cribl Lake are available on the [Invoices](#) page.

2. LAKE DATASETS

Organize different types of data stored in Cribl Lake, using Lake Datasets.

Cribl Lake comes with ready-to-use default Lake Datasets and lets you create your own Datasets.

You can have up to 200 Lake Datasets per Workspace. If you need more, reach out to [Cribl Support](#).

The data is stored as gzip-compressed JSON files.

Lake Datasets Retention

Every Lake Dataset stores data for a defined retention period.

For [built-in Datasets](#), the retention period is fixed to 10-30 days, depending on the Dataset.

For your own Datasets, you can [configure the retention period](#) depending on your needs.

How Lake Datasets Retention Is Calculated

The retention period of a Lake Dataset is based on the date on which data was uploaded and saved, not on the dates of individual stored events.

This distinction is important if you are uploading older events in a batch. Retention will then be calculated based on the date of the upload.

For example, if on the 1st Aug 2024 you upload a batch of data including events dated at 1st June 2024, and set the retention period to 1 year, the events will be deleted after 1st Aug 2025 (based on upload date), not in June 2025.

Built-in Lake Datasets

The following Lake Datasets are available by default in Cribl Lake:

Lake Dataset	Contains	Retention Period (Days)	Notes
default_logs	Logs from multiple sources.	30	
default_metrics	Metrics from multiple sources.	15	
default_spans	Distributed trace spans from multiple sources.	10	

Lake Dataset	Contains	Retention Period (Days)	Notes
default_events	Events from sources such as Kubernetes or a third-party API.	30	
cribl_metrics	Metrics from the Cloud Leader. Data is stored for 30 days free of charge.	30	Cannot be targeted by a Destination.
cribl_logs	Logs from the Cloud Leader. Data is stored for 30 days free of charge.	30	Cannot be targeted by a Destination.

Audit and Access Logs

Cribl Lake automatically collects audit and access logs, and metrics from the Cloud Leader Node (a node that manages the whole Cribl deployment). This data is stored in the `cribl_logs` and `cribl_metrics` Lake Datasets.

Because these Lake Datasets are internal, you can't use them as a target in the [Cribl Lake Destination](#).

You can't edit or delete built-in Lake Datasets.

2.1. MANAGE LAKE DATASETS

Create, edit, partition, datatype, and delete Lake Datasets.

This topic primarily covers Lake Datasets that you populate from Cribl Stream, using the [Cribl Lake Destination](#). However, for selected types of data, you have the option to create and populate a Lake Dataset using Cribl Lake [Direct Access](#).

You can also [send Cribl Search results to Cribl Lake](#) by using the Search [export](#) operator. Data ingested from Search to Lake will typically have field names and values pre-parsed, and such parsing is a prerequisite to achieving [Lakehouse](#) performance boost.

Create a New Lake Dataset

To create a new Lake Dataset:

1. In the Cribl Lake sidebar, select **Datasets**.
2. Select **Add Dataset** above the Dataset table.
3. Enter an **ID** for the new Dataset and, optionally, a **Description**.

The identifier can contain letters (lowercase or uppercase), numbers, and underscores, and can be no longer than 512 characters.

You can't change the identifier once a Dataset is created, so make sure it is meaningful and relevant to the data you plan to store in it.



The Lake Dataset identifier must be unique and must not match any existing [Cribl Search Datasets](#). Duplicate identifiers can cause problems when searching Cribl Lake data.

4. Define the **Retention period**. Cribl Lake will store data in the Dataset for the time duration you enter here. The allowed range is anywhere from 1 day to 10 years.



If you plan to link the Lake Dataset to a Lakehouse, we recommend setting the Dataset's retention period to 30 days or longer, as Lakehouses [cover](#) a rolling window of up to 30 days.

5. Select the **Data format**. JSON is the default. Parquet is a columnar storage format ideal for big data analytics (for tips on searching Parquet data, see [Parquet](#)). DDSS is available for a Dataset that you will populate using [Direct Access](#).
6. Optionally, select the [Lakehouse](#) you want to link to the Dataset. When you select a Lakehouse at this point, the Lake Dataset will become a [mirrored Dataset](#). (You cannot select this with a Dataset populated using [Direct Access](#).)
7. Optionally, in **Advanced Settings**, select up to three [Lake Partitions](#) for the Dataset.

8. If you are linking the Dataset to a Lakehouse, then in **Advanced Settings**, you can optionally select up to five **Lakehouse Indexed fields**.
9. Confirm with **Save**.

Datasets / **New Dataset**

ID* ⓘ
sampledataset

Description
My sample dataset

Retention period* ⓘ
1 Day 30 Days 60 Days 120 Days 1 Year 3 Years 5 Years 365 days

Data format* ⓘ
☒ JSON ☐ Parquet

Lakehouse ⓘ
testing Ready

Advanced Settings

Lake Partitions ⓘ
Add up to 3 partitioning fields to this Dataset to help retrieve data when searching or replaying.
1 hostname

Lakehouse Indexed fields ⓘ
Add up to 5 fields you intend to search frequently to improve query performance.
1 sourcetype

Cancel Save

Fill in a few fields and set up the retention period and you have a new Lake Dataset.

You will now be able to select this new Dataset by using its identifier in a [Cribl Lake Destination](#) or [Cribl Lake Collector](#).



Only [Organization Owners or Admins](#) can add (above) or modify (below) Lake Datasets.

Edit a Lake Dataset

To edit an existing Lake Dataset:

1. In the Cribl Lake sidebar, select **Datasets**.
2. Select the Dataset you want to edit and change the desired information, including description, [retention period](#), [partitions](#), and assigned [Lakehouse](#). You can't modify the identifier of an existing Lake Dataset.
3. Confirm with **Save**.

Change a Lake Dataset's Retention Period

If you change the retention period of an existing Lake Dataset, this affects all data in the Dataset.

If you increase the retention period, data **currently** in the Dataset will adopt the new retention period.



If you **decrease** the retention period, data older than the new time window will be lost. Make sure that this is your intention before you save the change.

Delete a Lake Dataset

You can delete a Lake Dataset either from the Dataset table, or from an individual Dataset's page.

To delete a Lake Dataset from the Dataset table:

1. In the Cribl Lake sidebar, select **Datasets**.
2. Select the check box next to the Dataset(s) you want to delete.
3. Select **Delete Selected Datasets**.

Alternatively, select the Dataset's row in the Dataset table, and then select **Delete**.

Scheduled Deletion

A Lake Dataset is not deleted instantly. Instead, once you select it for deletion, it will be marked with "Deletion in progress". At this point you can no longer edit it, or connect it to Collectors or Destinations. Data that is older than 24 hours will be removed at midnight UTC the following day, while more recent data may be deleted after that time. Once a Lake Dataset is marked for deletion, you are no longer charged for any data that it contains.



You cannot delete built-in Lake Datasets, or Lake Datasets that have any connected Collectors or Destinations.

Lake Partitions and Lakehouse-Indexed Fields

Each Lake Dataset can have up to three partitions configured, and, if you're using a Lakehouse, up to five Lakehouse-indexed fields, to speed up searching and replaying data from Cribl Lake.

A typical use case for applying partitions and indexed fields is to accelerate `hostname` or `sourcetype` if you are receiving data from multiple hosts or sources.

You can use partitions and indexed fields both in [Search queries](#) and in the filter for runs of the [Cribl Lake Collector](#).

Which Fields to Use as Partitions and Indexed Fields?

To ensure that search and replay work best, make sure you provide broader partitions or Lakehouse-indexed fields first. The order in which you configure partitions or indexed fields for a Lake Dataset influences the speed of search and replay of the data.

We also do not recommend partitions or indexed fields that:

- Contain PII (personally identifiable information).
- Contain objects.
- Have a name starting with an underscore (such as `_raw`).

For partitions, additionally avoid using fields that:

- Have high cardinality values.
- Are nullable (that is, can have the value of `null`).



Don't use `source` or `dataset` as partitions or Lakehouse-indexed fields. These are internal fields reserved for Cribl, and are not supported as partitions/indexed fields.

Datatypes for Lake Datasets

Lake Datasets, like regular Cribl Search Datasets, can be associated with [Datatypes](#). Datatypes help separate Dataset data into discrete events, timestamp them, and parse them as needed.

To configure Datatypes for a Lake Dataset:

1. On the top bar, select **Products**, and then select **Search**.
2. In the sidebar, select **Data**.
3. In the list of Datasets select your Lake Dataset.
4. Select the **Processing** tab.
5. In the **Datatypes** list, add desired Rulesets via the **Add Datatype Ruleset** button. (The order of Rulesets matters.)
6. When the list of Rulesets is complete, confirm with **Save**.



For more information about using Datatypes and Rulesets in searches, see [Cribl Search Datatypes](#).

3. SEARCH CRIBL LAKE

Search your Cribl Lake data with Cribl Search.

Cribl Lake appears as a preconfigured Cribl Search [Dataset Provider](#) called `cribl_lake`. This enables you to use Cribl Lake [Datasets](#) as Cribl Search Datasets and instantly start searching them.

Search a Lake Dataset

To access Cribl Search: From your Organization's top bar, select **Products**, then select **Search**. You can then [query](#) your Lake Datasets from Cribl Search.

Search a Lakehouse

You can speed up searching your Cribl Lake data by using a [Lakehouse](#). Once a Lake Dataset is assigned to a Lakehouse, searches against that Dataset will run significantly faster – as long as the search's whole range is contained within the Lakehouse's retention period.

If a query covers a time period wider than the data stored in a Lakehouse, Cribl Search will normally fall back to performing a regular search with corresponding latency. However, if the Lake Dataset has a retention period shorter than the Lakehouse, Cribl Search will still accelerate the search.

Verify Lakehouse Use

To verify whether a search successfully used a Lakehouse, take a look at the [tracking bar](#).

If the search failed to use a Lakehouse, the bar presents information about potential reasons. The reasons might include (as examples) the Lakehouse being disabled or misconfigured, or the query exceeding the time range of data stored in the Lakehouse.

Search Multiple Lakehouse Datasets

You can run a single query against multiple Lakehouse-assigned Datasets. For the query to execute at Lakehouse speed, all Datasets in the query must be Lakehouse-assigned, and your query must also meet one of these conditions:

- Include only operators from the following group: [cribl](#), [centralize](#), [extend](#), [extract](#), [foldkeys](#), [limit](#), [mv-expand](#), [pivot](#), [project](#), [project-away](#), [project-rename](#), [search](#), and [where](#).
- Or, if you use any other operators, insert the [centralize](#) or [limit](#) operator before them. (Result counts will be further constrained by [usage group](#) limits.)

If neither of the above conditions is met, or if your query includes non-Lakehouse Datasets, the query will run at standard speed.

Cribl Search Differences with Lakehouse

Executing Cribl Search queries against a Lakehouse-assigned Dataset changes some behavior and results, compared to executing the same queries without Lakehouse caching. For details, see [Lakehouse Search Differences](#).

Examples of Searching Cribl Lake

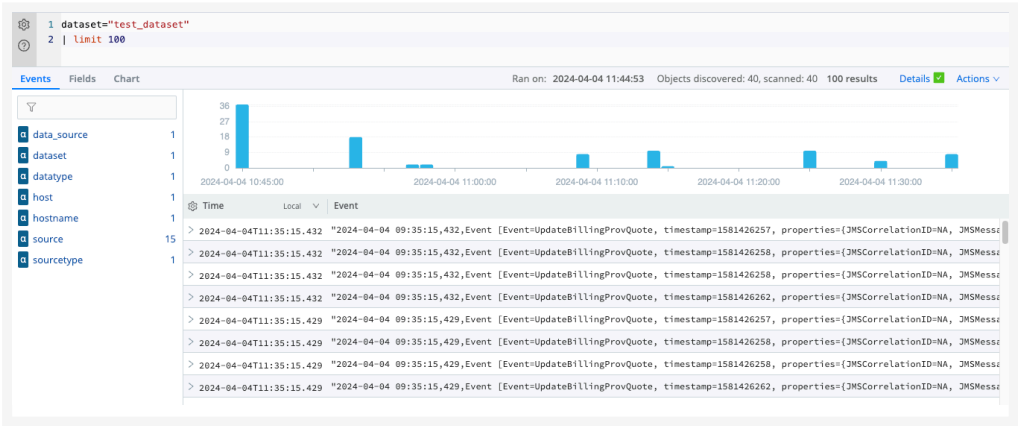
Use these examples as starting points for your own searches.

Basic Search into Cribl Lake

This search specifies the Dataset (`test_dataset`) and limits the number of results.

```
dataset="test_dataset"  
| limit 100
```

 Copy



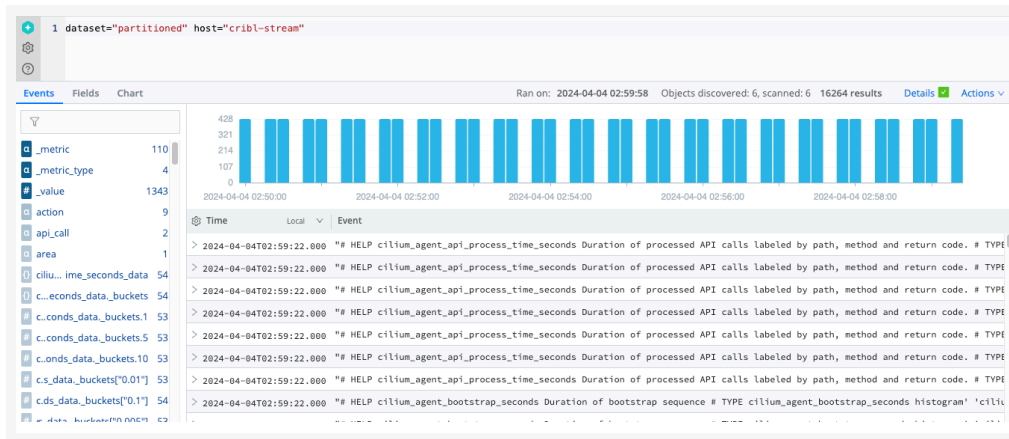
Sample Cribl Lake Search

Search Cribl Lake with a Partition

This search uses a [Lake partition](#) named `sourcetype` that is configured for the partitioned Dataset to speed up retrieval:

```
dataset="partitioned" host="cribl-stream"
```

 Copy



Sample Cribl Lake Search: Using partitions

Export Cribl Search Results to Cribl Lake

The [export](#) operator lets you export Cribl Search results to a Lake Dataset. You can later search this Dataset to extract relevant data from it.

An efficient way to search exported data is to provide the search job ID to the [where](#) operator:

```
dataset="exported_data"
| where source contains "1713177481843.9A0qxI"
```

Copy



You can find the search job ID in search details after running it, or in the [History](#) tab, in the **Search ID** column.

You can also label exported events using the [extend](#) operator and then include the added fields in your search. For example, during export you can include the user that performed the search:

```
dataset="cribl_search_sample"
| extend user = user()
| export to lake exported_data
```

Copy

Run in Cribl Search

You can then search for data by this user:

```
dataset="exported_data"  
| where user == "John Doe"
```

 Copy

4. LAKEHOUSES IN CRIBL LAKE

Search Cribl Lake faster with a Lakehouse.

To speed up searching data from Cribl Lake, you can assign individual Lake Datasets to a **Lakehouse**.

Lakehouses are a parallel option to regular Lake Datasets. When searching a Lakehouse, Cribl Search returns results with much shorter latency than when searching regular Lake Datasets.

When you assign a Dataset to a Lakehouse, data ingested into the Dataset is simultaneously sent to a cache storage system, which allows for quicker retrieval. Data sent to the Lakehouse [ages out](#) after up to 30 days, calculated from the time of ingest into the Lakehouse.



For examples of how Lakehouses can enable efficient storage and fast search of high-volume recent telemetry data, see: [VPC Flow Logs with a Lakehouse](#), [CloudTrail Logs with a Lakehouse](#), and [CloudFront Logs with a Lakehouse](#).

Lakehouse Availability

Lakehouses are available only in Cribl.Cloud Organizations on certain [plans](#).

Lakehouse Sizes

Each Lakehouse has a size. The size defines the amount of data that can be ingested into and stored in the Lakehouse. When sizing your Lakehouse, consider your expected data ingest for Lake Datasets associated with that Lakehouse. Then, select a size based on this estimate. (An undersized Lakehouse will be able to cache fewer days of data.)

The following sizes are available:

Size	Capacity (per day)
Small	600 GB
Medium	1200 GB
Large	2400 GB
XLarge	4800 GB
XXLarge	9600 GB

Size	Capacity (per day)
3XLarge (Contact Support)	14 TB
6XLarge (Contact Support)	28 TB

Exceeding Lakehouse Capacity

When data flowing into a Lakehouse exceeds the Lakehouse capacity, this will degrade both ingest speed and search speed. To prevent these impacts, try to avoid sustained excessive loads. If they occur, consider expanding the Lakehouse capacity, as covered in the next section.

Resize a Lakehouse

After creating a Lakehouse, if you decide you need less or more storage capacity, you can resize an existing Lakehouse.

To do so, on the **Lakehouses** page, select the existing Lakehouse and choose a different size.

When you resize a Lakehouse, reserve some time for its resouces to be reprovisioned. During this period, Lakehouse-cached searching will be unavailable. So any search against Lake Datasets connected to the Lakehouse will proceed at the speed of a regular search. (Cribl Search will also consume credits for CPU time, because these searches are not covered by Lakehouse flat billing.) After the resize operation is complete, your Lakehouse-cached data will be searchable again.

Lakehouse Retention

Each Lakehouse covers a rolling window of up to **30 days**, calculated **from the time of ingest into the Lakehouse**.

Lake Dataset Retention	Lakehouse Retention
1 day	1 day
30 days	30 days
120 days	30 days

Add a New Lakehouse

A Lake Dataset can be assigned to only one Lakehouse. However, a Lakehouse can have more than one Lake Dataset assigned to it.

To add a new Lakehouse, as an Organization Owner:

1. In the Cribl Lake sidebar, select **Lakehouses**.
2. Select **Add Lakehouse**.
3. Enter an **ID** for the new Lakehouse and, optionally, a **Description**.
4. Define the **Size** of the Lakehouse, based on expected ingest volume.
 - You can change the size later.
 - There are two larger Lakehouse sizes: 3XLarge and 6XLarge. To use one of these sizes, [Contact Support](#).

The new Lakehouse will be fully active after it has been provisioned, which might take up to an hour.

Lakehouses / New Lakehouse	
ID*	
my_lakehouse	
Description	
Enter description	
Lakehouse Size*	
Name	Capacity (GB per day)
☐ Small	600
☒ Medium	1200
☐ Large	2400
☐ XLarge	4800
☐ XXLarge	9600
Cancel Save	

Select an initial size for the new Lakehouse

Link a Lake Dataset to a Lakehouse

You can link one or more Lake Datasets to each Lakehouse. To do so, you need to be an [Organization Admin](#) or [Lake Admin](#).

Linking Datasets is possible only after the Lakehouse has been fully provisioned.

You can link Datasets in two ways, influencing how the Dataset will behave in relation to the Lakehouse:

- [Link an already existing Lake Dataset](#), to only cache data ingested into the Dataset in the future.
- [Link a new Lake Dataset](#), to mirror the Dataset's contents.

Link an Existing Lake Dataset to a Lakehouse

To link an existing Lake Dataset to a Lakehouse:

1. In the sidebar, select **Datasets**.

2. On the resulting **Datasets** page, select the Lake Dataset you want to assign to a Lakehouse.
3. In the **Lakehouse** drop-down, select the Lakehouse you want to assign the Dataset to.
4. To get the most out of the Lakehouse, set the Dataset's **Retention period** to **30 days** or longer, as Lakehouses [cover](#) a rolling window of up to 30 days.

The screenshot shows the configuration page for a dataset named 'cribl_logs'. The 'ID*' field contains 'cribl_logs'. The 'Description' field contains 'Internal logs from Cribl deployments'. The 'Retention period*' is set to '30 days' on a slider ranging from '1 Day' to '5 Years'. The 'Data format*' is set to 'JSON'. The 'Lakehouse' dropdown menu is open, showing a search bar with 'None selected' and a list of three datasets: 'dataset1', 'another_dataset', and 'my_dataset', each with a 'Ready' button next to it.

Link Lakehouses to Lake Datasets

When you assign an existing Lake Dataset to a Lakehouse, only data newly ingested to that Dataset will be sent to the Lakehouse.



If you link an **empty** existing Lake Dataset to a Lakehouse, it will behave like a [mirrored Dataset](#).

Link a New Lake Dataset to a Lakehouse

To link a Lake Dataset to a Lakehouse during its creation:

1. In the sidebar, select **Datasets**.
2. Select **Add Dataset** above the Dataset table.
3. Configure the Dataset as described in [Create a New Lake Dataset](#).
4. In the **Lakehouse** dropdown, select the Lakehouse you want to link to.

Assigning a Lake Dataset to a Lakehouse during creation results in a mirrored Dataset.

Mirrored Lake Datasets

When you assign a Lake Dataset to a Lakehouse **while creating** the Dataset, the Lakehouse will mirror the Dataset's contents.

Data sent to a mirrored Dataset will be ingested into the Lakehouse regardless of the `event_time` of the event. Cribl Search will always use the Lakehouse when searching a mirrored Dataset, unless you explicitly turn

it off by setting the [Lakehouse](#) option to off.

Insert Data into a Lakehouse

To populate a Lakehouse, see [Prepare Data for Use with Lakehouse](#).

Delete a Lakehouse

To delete a Lakehouse:

1. In the sidebar, select **Lakehouses**.
2. Select the check box next to the Lakehouse(s) you want to delete.
3. Select **Delete Selected Lakehouses**.

A Lakehouse is not deleted instantly. Instead, once you select the Lakehouse for deletion, it will be marked with “Deletion in progress”. At this point, you can no longer edit the Lakehouse, nor connect it to Lake Datasets.

Data that is older than 24 hours will be removed at midnight UTC the following day, while more-recent data might be deleted after that time. Once a Lakehouse is marked for deletion, you are no longer charged to maintain any data that it contains.



When you remove a Lakehouse, its data will still be available to be searched and replayed from its standard Lake Dataset.

Lakehouse Status

A Lakehouse can have one of the following statuses:

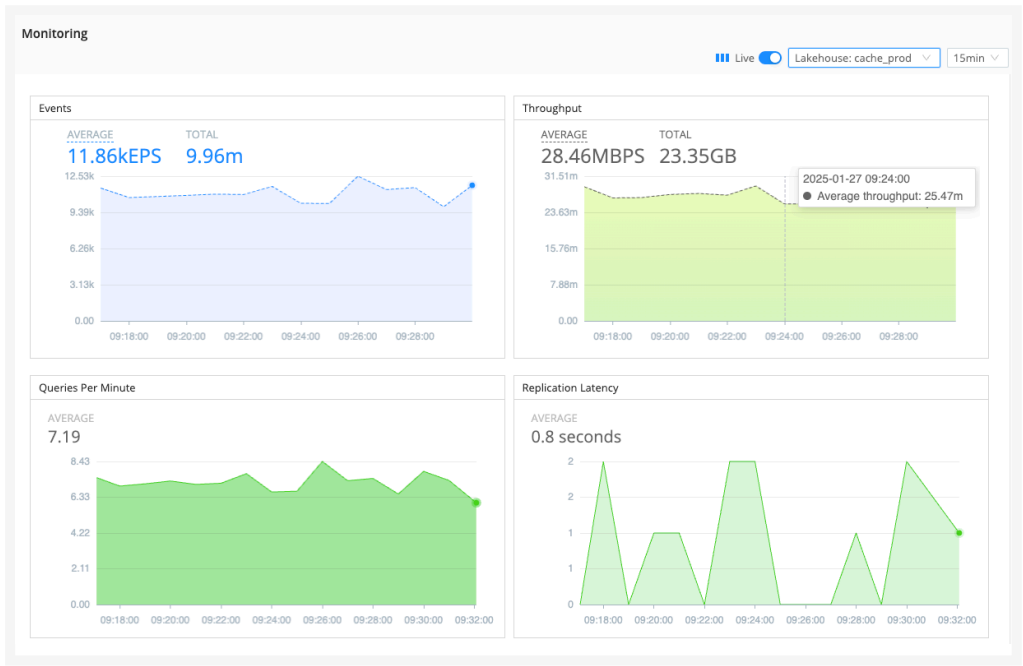
Status	Description
Ready	Lakehouse can be linked to Lake Datasets and used. You can resize it.
Provisioning	Lakehouse is being prepared, you cannot resize it or link it to Lake Datasets.
Delayed	Provisioning the Lakehouse has encountered an issue, but it will move to Ready status when the issue is resolved. While the Lakehouse is Delayed, you can resize it, but you can’t link it to Datasets.
Terminated	Lakehouse has been marked for deletion .

Monitor Lakehouse Usage

To control your usage of Lakehouses, and to make decisions about their potential resizing, you can display charts that present data per Lakehouse.

Select the sidebar's **Monitoring** link. On the resulting page, use the drop-down at the upper right to select among Lakehouses. You'll see the following charts.

- **Events:** Number of events ingested into the Lakehouse.
- **Throughput:** Number of bytes ingested into the Lakehouse.
- **Queries per Minute:** Number of queries performed on the Lakehouse.
- **Replication Latency:** Latency of ingesting data into the Lakehouse.



Charts to monitor performance per Lakehouse

On the **Lakehouses** page, you can see summary data for all configured Lakehouses. **Ingestion Rate** measures the past 24 hours, compared with each Lakehouse's configured capacity.

Max Throughput graphically compares actual throughput to a guideline representing the recommended maximum, again based on the configured capacity.

Lakehouses						
What are Lakehouses? Lakehouses accelerate recent data for a faster search experience. You assign one or more Cribl Lake Datasets to a Lakehouse to automatically accelerate new data sent to that Dataset.						
Filter Lakehouses						
Add Lakehouse						
ID & Description	Status	Datasets	Lakehouse Size	Max Throughput	Ingestion Rate (24h)	Actions
Ogopogo	Delayed	N/A	Small	N/A	N/A	...
test_lakehaus	Ready	3 datasets	Small		1.69 GB / 600 GB	...
coateshouse Haus Of Coates	Ready	cribl_logs	Small		0.87 GB / 600 GB	...
Weather_Lakehouse	Ready	weather_data	Small		< 0.01 GB / 600 GB	...
keanu_bullock	Ready	MPF-1747163322	Small	No Results		...

4.1. STREAMLINE VPC FLOW LOGS ANALYSIS WITH A LAKEHOUSE

Use a Cribl Lakehouse to efficiently store, and quickly analyze, high-volume recent telemetry data.

This is a straightforward, end-to-end example of how to ingest AWS [VPC Flow Logs](#) into a Lakehouse, and how to then use [Cribl Search](#) to periodically analyze only specific data that is relevant for threat-hunting or security investigations.

Especially as data volume increases, this can be a cost-effective alternative to bulk-forwarding data to a traditional SIEM system. This example walks you through sequential setup in AWS, Cribl Stream, Cribl Lake, and Cribl Search. The broad steps are:

1. [Collect](#) and stage VPC Flow Logs in AWS.
2. [Shape](#) the data, using a Cribl Stream Parser.
3. [Create](#) a Cribl Lake Dataset to store your parsed data.
4. [Route](#) the parsed data to the new Lake Dataset.
5. [Configure](#) a Lakehouse, to support low-latency analytics in Cribl Search.
6. [Analyze](#) relevant data, using Cribl Search ad hoc and scheduled queries.
7. [Visualize](#) your data, using Cribl Search predefined or custom Dashboards.



Beyond this specific example, you can use a similar workflow to efficiently store and analyze many other types of logs, by substituting a different Cribl Stream [Parser](#) or configuring your own.

Collect and Stage VPC Flow Logs

Your first steps take place in the Amazon VPC console or EC2 console. If you're not already collecting VPC Flow Logs, the Amazon [Logging IP Traffic Using VPC Flow Logs](#) documentation provides specific instructions.

1. [Set up](#) AWS VPC Flow logging within your AWS account.
2. Configure [writing](#) the VPC Flow Logs to an Amazon S3 bucket.
3. [Enable permissions](#) for Cribl Stream to read from the S3 bucket.

Shape the Data

Next, use Cribl Stream to ingest your VPC Flow Logs data, and to parse the named fields that Lakehouse requires. (You'll want to configure all Stream steps within your Cribl.Cloud Organization, to enable [routing to Cribl Lake](#) later.)

1. Configure a Cribl Stream [Amazon S3 Source](#) to continuously collect (stream) data from your S3 bucket.
2. Create or adapt a simple Cribl Stream [Pipeline](#) that will process this data.
3. Add a single [Parser Function](#) to the Pipeline.
4. In the Parser, accept default Extract mode, then set the **Library** to AWS VPC Flow Logs. As shown below, this will automatically set the **Type** to Extended Log File Format.
5. Save the Pipeline.

The screenshot shows the 'Parser' configuration window in Cribl Stream. The window has a title bar with a toggle switch (turned on), the word 'Parser', and the value 'true'. Below the title bar, there are several sections:

- Filter**: A text input field containing 'JS true'.
- Description**: A text input field with the placeholder 'Enter a description'.
- Final**: A toggle switch (turned off).
- Operation mode***: A dropdown menu set to 'Extract'.
- Type***: A dropdown menu set to 'Extended Log File Format'.
- Library**: A dropdown menu set to 'AWS VPC Flow Logs'.
- Source field**: A text input field containing '_raw'.
- Destination field**: A text input field with the placeholder 'Destination field name'.
- List of fields**: A list of fields with checkboxes, including 'version', 'account_id', 'interface_id', 'srcaddr', 'dstaddr', 'srcport', 'dstport', 'protocol', 'packets', 'bytes', 'start', 'end', 'action', and 'log_status'.
- Fields to keep**: A text input field with the placeholder 'Enter field names'.
- Fields to remove**: A text input field with the placeholder 'Enter field names'.
- Fields filter expression**: A text input field containing 'JS Enter field filter expression'.

Parser configuration for VPC Flow Logs



You might choose to expand this Pipeline to add Tags, or to add other predefined Cribl Stream [Functions](#) to shape or narrow your data in specific ways. This scenario presents the simplest scenario to parse your data for Cribl Lake.

The Cribl [AWS VPC Flow for Security Teams](#) Pack provides sample Pipelines that you can adapt and add to Cribl Stream. For usage details, see [Cribl Stream for InfoSec: VPC Flow Logs – Reduce or Enrich? Why Not Both?](#) You can also use the [Cribl Copilot Editor](#) (AI) feature to build a Pipeline from a natural-language description of the processing you want.

Create a Cribl Lake Dataset

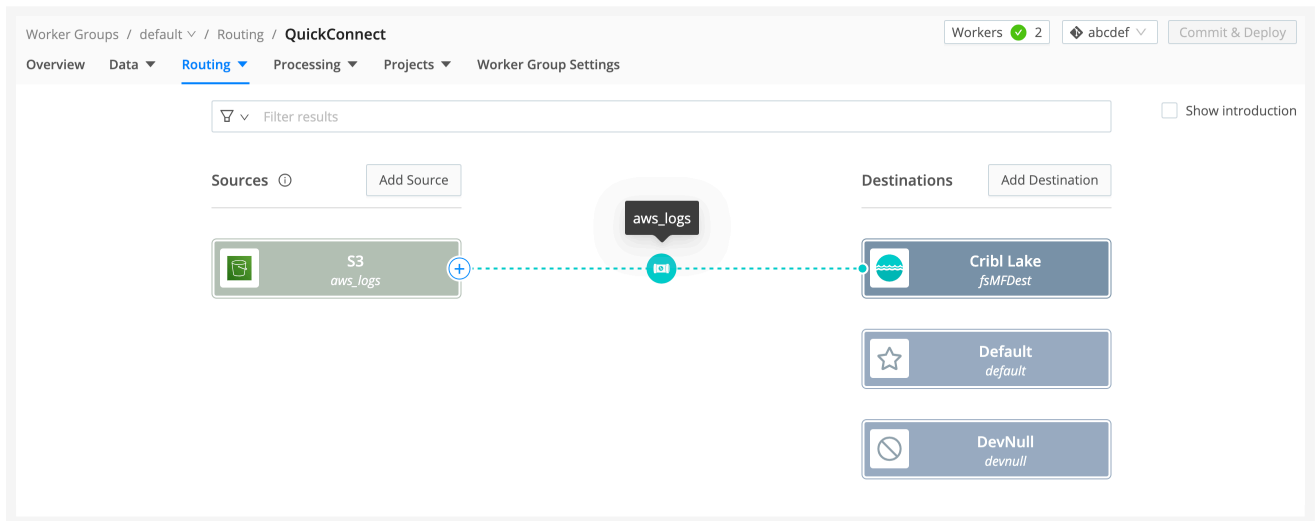
In Cribl Lake, [create](#) a Lake Dataset to store your shaped data.

Lakehouse acceleration covers a rolling 30-day retrospective time window, so configure your Lake Dataset with at least a 30-day retention period – or longer, depending on your organization’s retention policy and compliance needs.

Route the Parsed Data to Cribl Lake

Back in Cribl Stream, route your shaped data to your Lake Dataset.

1. Add a [Cribl Lake Destination](#).
2. Within that Destination, select the Lake Dataset you’ve just created, and save the Destination.
3. You can use either QuickConnect or Data Routes to connect your S3 Source to your Lake Destination through the [Pipeline](#) you configured earlier. (The Destination topic linked above covers both options.)



Example QuickConnect configuration (data not yet flowing)

Configure a Lakehouse For Low-Latency Analytics

With data flow now established from S3 to Cribl Lake, assign the Lake Dataset to a Lakehouse to enable fast analytics.

1. [Add](#) a Lakehouse, [sized](#) to match your expected daily ingest volume.
2. When the Lakehouse is provisioned, [assign](#) your VPC Flow Logs Dataset to the Lakehouse.

Analyze Relevant Data

In [Cribl Search](#), define queries to extract relevant data from your VPC Flow Logs. You can search up to 30 days of data at Lakehouse speed. Here is a sample query to retrieve and present rejected connections by source

address:

```
dataset="<your_lakehouse_dataset>"  
| action="REJECT"  
| summarize by srcaddr
```

 Copy

You can [schedule your searches](#) to run on defined intervals. Once a query is scheduled, you can configure corresponding alerts (Search [Notifications](#)) to trigger on custom conditions that you define.

Here is a sample query you could schedule to generate an hourly report that tracks traffic by source and destination:

```
dataset = "<your_lakehouse_dataset>" srcaddr="*" dstaddr="*"   
| summarize count() by srcaddr, dstaddr   
| sort by count desc
```

 Copy



If you're moving high-volume log analysis from a traditional SIEM to Cribl, you might need to adapt existing queries to the KQL language that Cribl Search uses. See our [Build a Search](#) overview and our [Common Query Examples](#).

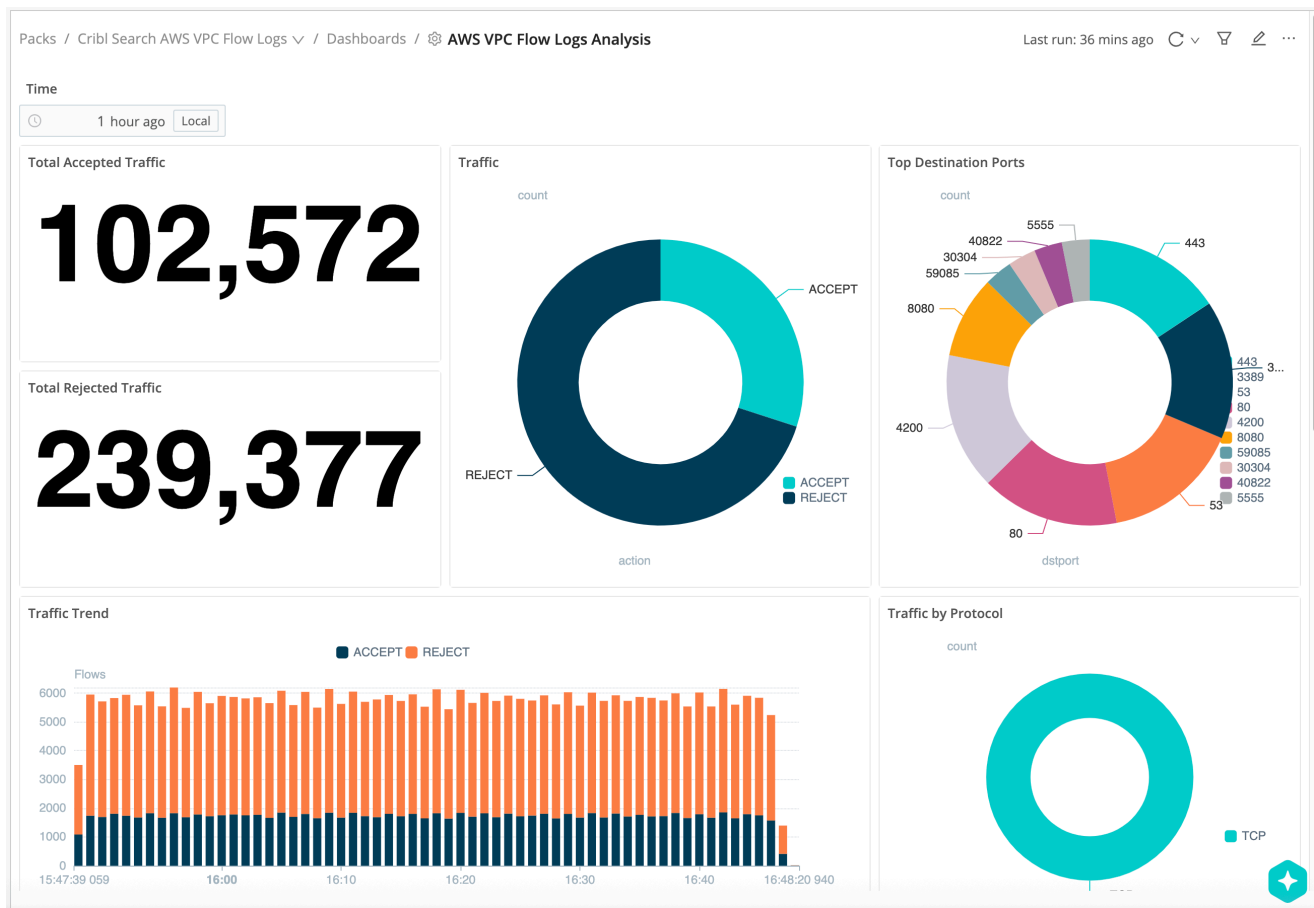
You can also [Write Queries Using Cribl Copilot](#), enabling Cribl AI to suggest KQL queries from your natural-language prompts. If you already have searches in another system of analysis, try asking Copilot to translate them to KQL.

The `cribl_search_sample` Dataset, built into Search, contains VPC Flow Log events. You can use this Dataset to experiment with queries before you enable continuous data flow and analysis.

Visualize Your Analyzed Data

In Cribl Search, you can build flexible [Dashboards](#) to visualize your analytics.

As a starting point, you can [import](#) the [Cribl Search AWS VPC Flow Logs](#) Pack. It contains a sample **AWS VPC Flow Logs Analysis** Dashboard that displays traffic volume, traffic accepted and rejected (both time-series and cumulative), traffic by protocol, and top destination ports.



VPC Flow Logs Dashboard



Cribl AI can automatically suggest visualizations for your Dataset, and can build visualizations from your natural-language prompts. For these options, see [Add Visualizations Using Cribl Copilot](#).

Next Steps

To refine your queries and visualizations, see our Cribl Search specialized topics on:

- [Searching Amazon S3](#)
- [Optimize Cribl Search](#)
- [Create a Dashboard](#)
- [Edit a Dashboard](#)

4.2. STREAMLINE CLOUDTRAIL LOGS ANALYSIS WITH A LAKEHOUSE

Use a Cribl Lakehouse to efficiently store, and quickly analyze, high-volume recent telemetry data.

This is a straightforward, end-to-end example of how to ingest [AWS CloudTrail log files](#) into a Lakehouse, and how to use [Cribl Search](#) to periodically analyze only specific data that is relevant for threat-hunting or security investigations.

Especially as data volume increases, this can be a cost-effective alternative to bulk-forwarding data to a traditional SIEM system. This example walks you through sequential setup in AWS, Cribl Stream, Cribl Lake, and Cribl Search. The broad steps are:

1. [Collect](#) and stage CloudTrail logs in AWS.
2. [Shape](#) the data, using a Cribl Stream Parser.
3. [Create](#) a Cribl Lake Dataset to store your parsed data.
4. [Route](#) the parsed data to the new Lake Dataset.
5. [Configure](#) a Lakehouse, to support low-latency analytics in Cribl Search.
6. [Analyze](#) relevant data, using Cribl Search ad hoc and scheduled queries.
7. [Visualize](#) your data, using Cribl Search predefined or custom Dashboards.



Beyond this specific example, you can use a similar workflow to efficiently store and analyze many other types of logs, by substituting a different Cribl Stream [Parser](#) or configuring your own.

Collect and Stage CloudTrail Logs

Your first steps take place in the Amazon Management Console. If you're not already collecting CloudTrail logs, the Amazon [What Is AWS CloudTrail?](#) documentation provides specific instructions.

1. [Set up](#) trails within your AWS account to log events of interest.
2. Configure [writing](#) the CloudTrail logs to an Amazon S3 bucket.
3. [Enable permissions](#) for Cribl Stream to read from the S3 bucket.

Shape the Data

Next, use Cribl Stream to ingest your CloudTrail data, and to parse the named fields that Lakehouse requires. (You'll want to configure all Stream steps within your Cribl.Cloud Organization, to enable [routing to Cribl Lake](#) later.)

1. Configure a Cribl Stream [Amazon S3 Source](#) to continuously collect (stream) data from your S3 bucket.
2. Create or adapt a simple Cribl Stream [Pipeline](#) that will process this data.
3. Add a single [Parser Function](#) to the Pipeline.
4. In the Parser, accept default Extract mode, then set the **Library** to AWS VPC Flow Logs (this library can handle CloudTrail logs, too). As shown below, this will automatically set the **Type** to Extended Log File Format.
5. Save the Pipeline.

The screenshot shows the 'Parser' configuration window in Cribl Stream. The window has a title bar with a toggle switch (turned on), the word 'Parser', and the status 'true'. Below the title bar, there's a 'Filter' section with a text input containing 'JS true' and a 'Help' link. A 'Description' section has a text input with the placeholder 'Enter a description'. A 'Final' section has a toggle switch (turned off). The 'Operation mode*' section has a dropdown menu set to 'Extract'. The 'Type*' section has a dropdown menu set to 'Extended Log File Format'. The 'Library' section has a dropdown menu set to 'AWS VPC Flow Logs'. The 'Source field' section has a text input containing '_raw'. The 'Destination field' section has a text input with the placeholder 'Destination field name'. The 'List of fields' section shows a grid of field names: version, account_id, interface_id, srcaddr, dstaddr, srcport, dstport, protocol, packets, bytes, start, end, action, and log_status. The 'Fields to keep' section has a text input with the placeholder 'Enter field names'. The 'Fields to remove' section has a text input with the placeholder 'Enter field names'. The 'Fields filter expression' section has a text input containing 'JS Enter field filter expression'.

Parser configuration for CloudTrail logs



You might choose to expand this Pipeline to add Tags, or to add other predefined Cribl Stream [Functions](#) to shape or narrow your data in specific ways. This scenario presents the simplest scenario to parse your data for Cribl Lake.

The [Cribl Pack for AWS CloudTrail Data Collection](#) provides sample Pipelines that you can adapt and add to Cribl Stream. You can also use the [Cribl Copilot Editor](#) (AI) feature to build a Pipeline from a natural-language description of the processing you want.

Create a Cribl Lake Dataset

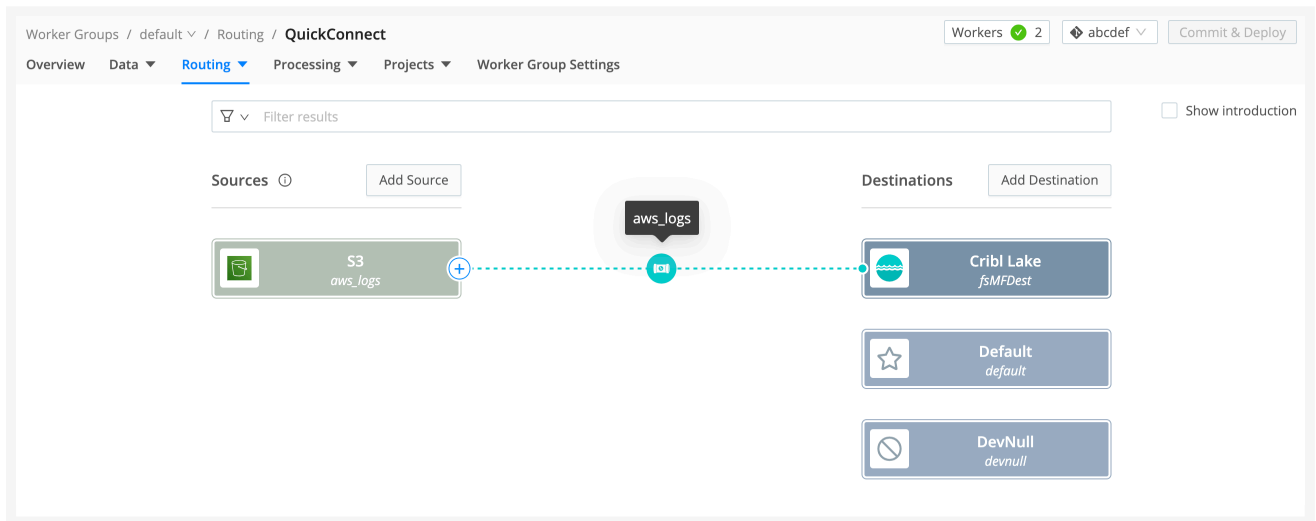
In Cribl Lake, [create](#) a Lake Dataset to store your shaped data.

Lakehouse acceleration covers a rolling 30-day retrospective time window, so configure your Lake Dataset with at least a 30-day retention period – or longer, depending on your organization’s retention policy and compliance needs.

Route the Parsed Data to Cribl Lake

Back in Cribl Stream, route your shaped data to your Lake Dataset.

1. Add a [Cribl Lake Destination](#).
2. Within that Destination, select the Lake Dataset you’ve just created, and save the Destination.
3. You can use either QuickConnect or Data Routes to connect your S3 Source to your Lake Destination through the [Pipeline](#) you configured earlier. (The Destination topic linked above covers both options.)



Example QuickConnect configuration (data not yet flowing)

Configure a Lakehouse For Low-Latency Analytics

With data flow now established from S3 to Cribl Lake, assign the Lake Dataset to a Lakehouse to enable fast analytics.

1. [Add](#) a Lakehouse, [sized](#) to match your expected daily ingest volume.
2. When the Lakehouse is provisioned, [assign](#) your CloudTrail Lake Dataset to the Lakehouse.

Analyze Relevant Data

In [Cribl Search](#), define queries to extract relevant data from your CloudTrail data. You can search up to 30 days of data at Lakehouse speed. Here is a sample query to retrieve and present rejected connections by source

address:

```
dataset="<your_lakehouse_dataset>"  
| action="REJECT"  
| summarize by srcaddr
```

 Copy

You can [schedule your searches](#) to run on defined intervals. Once a query is scheduled, you can configure corresponding alerts (Search [Notifications](#)) to trigger on custom conditions that you define.

Here is a sample query you could schedule to generate an hourly report that tracks traffic by source and destination:

```
dataset = "<your_lakehouse_dataset>" srcaddr="*" dstaddr="*"   
| summarize count() by srcaddr, dstaddr   
| sort by count desc
```

 Copy



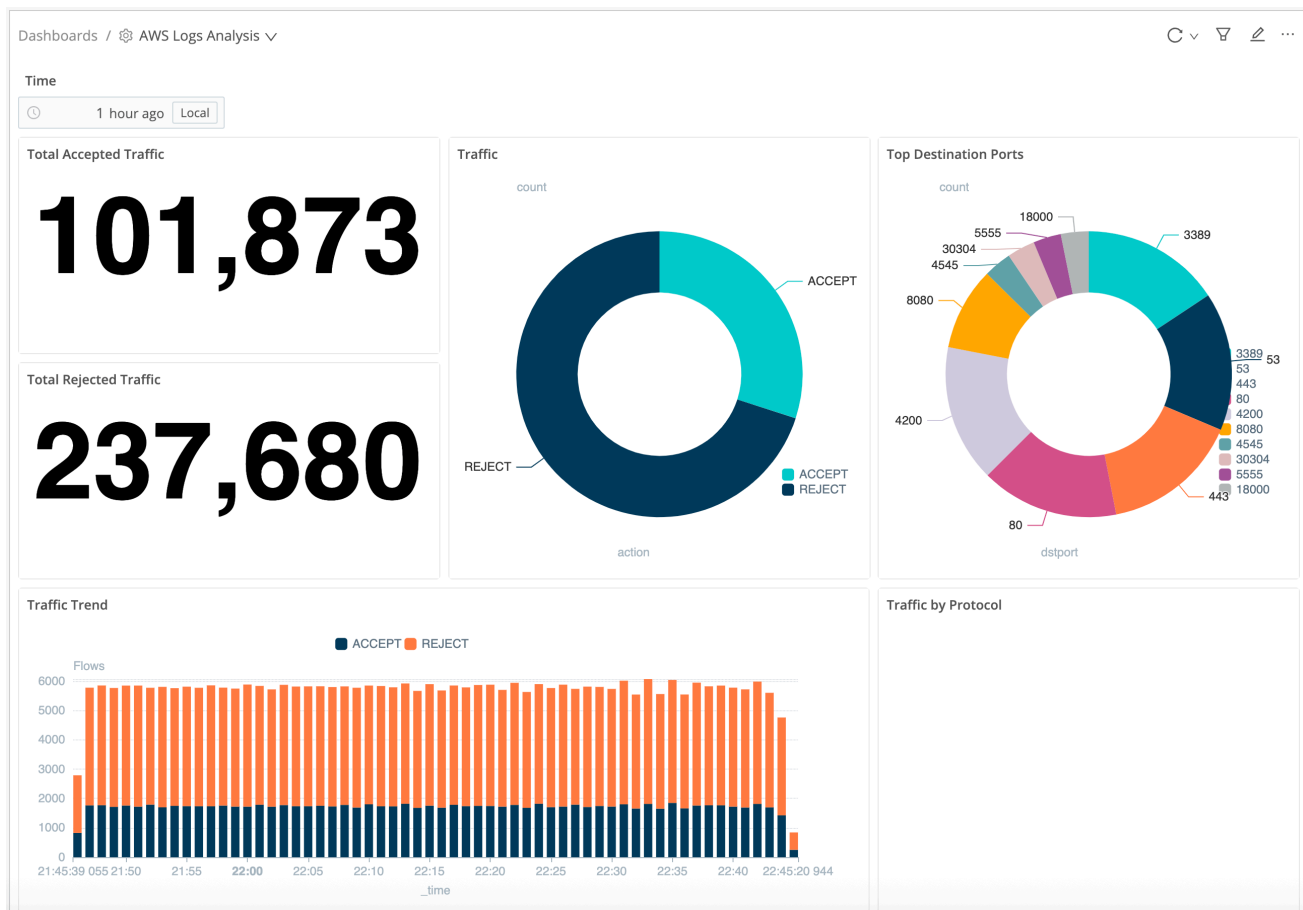
If you're moving high-volume log analysis from a traditional SIEM to Cribl, you might need to adapt existing queries to the KQL language that Cribl Search uses. See our [Build a Search](#) overview and our [Common Query Examples](#).

You can also [Write Queries Using Cribl Copilot](#), enabling Cribl AI to suggest KQL queries from your natural-language prompts. If you already have searches in another system of analysis, try asking Copilot to translate them to KQL.

The `cribl_search_sample` Dataset, built into Search, contains VPC Flow Log events. You can use this Dataset to experiment with queries before you enable continuous data flow and analysis.

Visualize Your Analyzed Data

In Cribl Search, you can build flexible [Dashboards](#) to visualize your analytics.



Sample Cribl Search Dashboard



Cribl AI can automatically suggest visualizations for your Dataset, and can build visualizations from your natural-language prompts. For these options, see [Add Visualizations Using Cribl Copilot](#).

Next Steps

To refine your queries and visualizations, see our Cribl Search specialized topics on:

- [Searching Amazon S3](#)
- [Optimize Cribl Search](#)
- [Create a Dashboard](#)
- [Edit a Dashboard](#)

4.3. STREAMLINE CLOUDFRONT LOGS ANALYSIS WITH A LAKEHOUSE

Use a Cribl Lakehouse to efficiently store, and quickly analyze, high-volume recent telemetry data.

This is a straightforward, end-to-end example of how to ingest [Amazon CloudFront log files](#) into a Lakehouse, and then use [Cribl Search](#) to periodically analyze only specific data that is relevant for threat-hunting or security investigations.

Especially as data volume increases, this can be a cost-effective alternative to bulk-forwarding data to a traditional SIEM system. This example walks you through sequential setup in Amazon, Cribl Stream, Cribl Lake, and Cribl Search. The broad steps are:

1. [Collect](#) and stage CloudFront logs in Amazon.
2. [Shape](#) the data, using a Cribl Stream Parser.
3. [Create](#) a Cribl Lake Dataset to store your parsed data.
4. [Route](#) the parsed data to the new Lake Dataset.
5. [Configure](#) a Lakehouse, to support low-latency analytics in Cribl Search.
6. [Analyze](#) relevant data, using Cribl Search ad hoc and scheduled queries.
7. [Visualize](#) your data, using Cribl Search predefined or custom Dashboards.



Beyond this specific example, you can use a similar workflow to efficiently store and analyze many other types of logs, by substituting a different Cribl Stream [Parser](#) or configuring your own.

Collect and Stage CloudFront Logs

Your first steps take place in Amazon interfaces. If you're not already collecting CloudFront logs, the Amazon [CloudFront and Edge Function Logging](#) documentation provides specific instructions.

1. In your Amazon account, set up CloudFront logging options of interest.
2. Configure [writing](#) the CloudFront logs to an Amazon S3 bucket.
3. [Enable permissions](#) for Cribl Stream to read from the S3 bucket.

Shape the Data

Next, use Cribl Stream to ingest your CloudFront data, and to parse the named fields that Lakehouse requires. (You'll want to configure all Stream steps within your Cribl.Cloud Organization, to enable [routing to Cribl Lake](#) later.)

1. Configure a Cribl Stream [Amazon S3 Source](#) to continuously collect (stream) data from your S3 bucket.
2. Create or adapt a simple Cribl Stream [Pipeline](#) that will process this data.
3. Add a single [Parser Function](#) to the Pipeline.
4. In the Parser, accept default **Extract** mode, then set the **Library** to **AWS CloudFront**. As shown below, this will automatically set the **Type** to **Delimited values**.
5. Save the Pipeline.

The screenshot shows the 'Parser' configuration window in Cribl Stream. The title bar indicates the parser is named 'true' and is currently active. The configuration includes the following sections:

- Filter:** Set to 'JS true'.
- Description:** A text input field with the placeholder 'Enter a description'.
- Final:** A toggle switch currently turned off.
- Operation mode*:** A dropdown menu set to 'Extract'.
- Type*:** A dropdown menu set to 'Delimited values'.
- Library:** A dropdown menu set to 'AWS CloudFront'.
- Delimiter:** A text input field containing '\t'.
- Quote char:** A text input field containing '" '.
- Escape char:** A text input field containing '\\ '.
- Null value:** A text input field containing '-'.
- Source field:** A text input field containing '_raw'.
- Destination field:** A text input field containing 'Destination field name'.
- List of fields:** A collection of predefined fields for CloudFront logs, each with a toggle switch. The fields include: 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, cs_cookie, x_edge_result_type, x_edge_request_id, x_host_header, cs_protocol, cs_bytes, time_taken, x_forwarded_for, ssl_protocol, ssl_cipher, x_edge_response_result_type, cs_protocol_version, fle_status, and fle_encrypted_fields.
- Fields to keep:** A text input field with the placeholder 'Enter field names'.
- Fields to remove:** A text input field with the placeholder 'Enter field names'.
- Fields filter expression:** A text input field with the placeholder 'JS Enter field filter expression'.

Parser configuration for CloudFront logs



You might choose to expand this Pipeline to add Tags, or to add other predefined Cribl Stream [Functions](#) to shape or narrow your data in specific ways. This scenario presents the simplest scenario to parse your data for Cribl Lake.

You can also use the [Cribl Copilot Editor](#) (AI) feature to build a Pipeline from a natural-language description of the processing you want.

Create a Cribl Lake Dataset

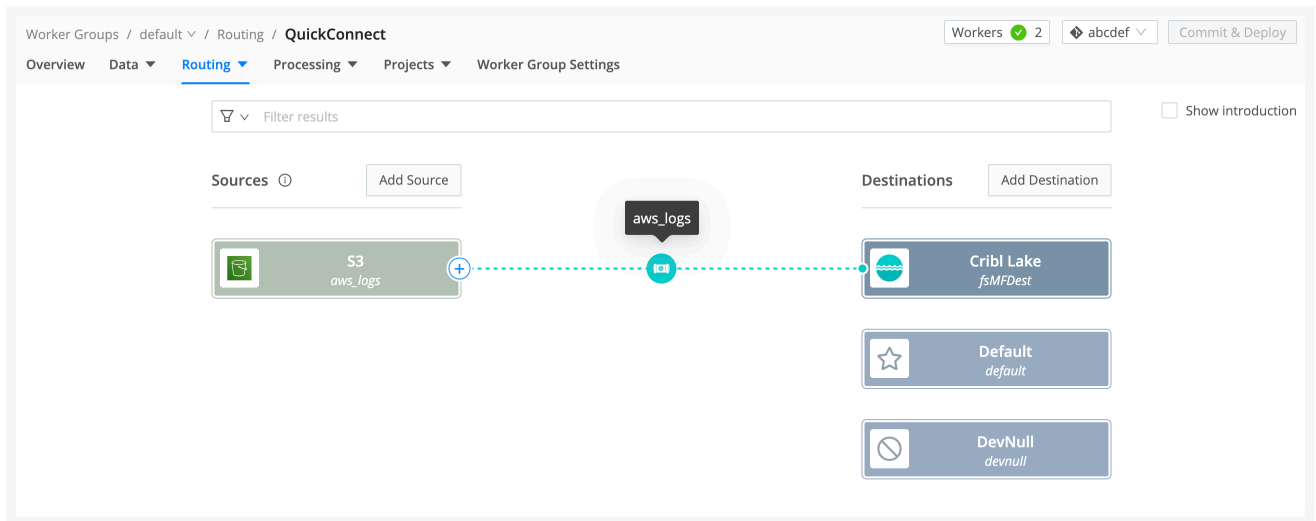
In Cribl Lake, [create](#) a Lake Dataset to store your shaped data.

Lakehouse acceleration covers a rolling 30-day retrospective time window, so configure your Lake Dataset with at least a 30-day retention period – or longer, depending on your organization’s retention policy and compliance needs.

Route the Parsed Data to Cribl Lake

Back in Cribl Stream, route your shaped data to your Lake Dataset.

1. Add a [Cribl Lake Destination](#).
2. Within that Destination, select the Lake Dataset you’ve just created, and save the Destination.
3. You can use either QuickConnect or Data Routes to connect your S3 Source to your Lake Destination through the [Pipeline](#) you configured earlier. (The Destination topic linked above covers both options.)



Example QuickConnect configuration (data not yet flowing)

Configure a Lakehouse For Low-Latency Analytics

With data flow now established from S3 to Cribl Lake, assign the Lake Dataset to a Lakehouse to enable fast analytics.

1. [Add](#) a Lakehouse, [sized](#) to match your expected daily ingest volume.
2. When the Lakehouse is provisioned, [assign](#) your CloudFront Lake Dataset to the Lakehouse.

Analyze Relevant Data

In [Cribl Search](#), define queries to extract relevant data from your CloudFront data. You can search up to 30 days of data at Lakehouse speed. Here is a sample query to retrieve and present rejected connections by source

address:

```
dataset="<your_lakehouse_dataset>"  
| action="REJECT"  
| summarize by srcaddr
```

 Copy

You can [schedule your searches](#) to run on defined intervals. Once a query is scheduled, you can configure corresponding alerts (Search [Notifications](#)) to trigger on custom conditions that you define.

Here is a sample query you could schedule to generate an hourly report that tracks traffic by source and destination:

```
dataset = "<your_lakehouse_dataset>" srcaddr="*" dstaddr="*"   
| summarize count() by srcaddr, dstaddr   
| sort by count desc
```

 Copy



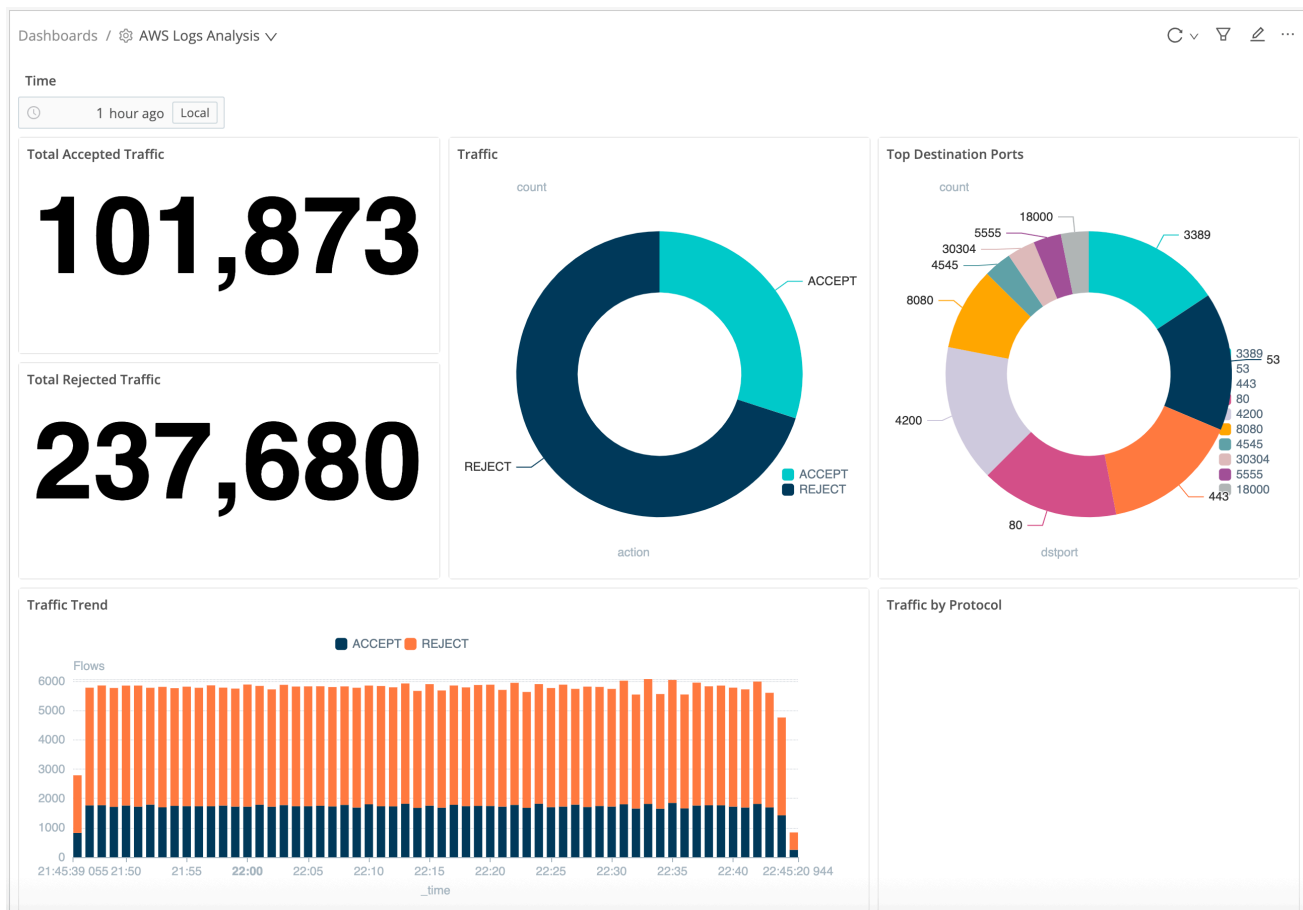
If you're moving high-volume log analysis from a traditional SIEM to Cribl, you might need to adapt existing queries to the KQL language that Cribl Search uses. See our [Build a Search](#) overview and our [Common Query Examples](#).

You can also [Write Queries Using Cribl Copilot](#), enabling Cribl AI to suggest KQL queries from your natural-language prompts. If you already have searches in another system of analysis, try asking Copilot to translate them to KQL.

The `cribl_search_sample` Dataset, built into Cribl Search, contains VPC Flow Logs events. You can use this Dataset to experiment with queries before you enable continuous data flow and analysis.

Visualize Your Analyzed Data

In Cribl Search, you can build flexible [Dashboards](#) to visualize your analytics.



Sample Cribl Search Dashboard



Cribl AI can automatically suggest visualizations for your Dataset, and can build visualizations from your natural-language prompts. For these options, see [Add Visualizations Using Cribl Copilot](#).

Next Steps

To refine your queries and visualizations, see our Cribl Search specialized topics on:

- [Searching Amazon S3](#)
- [Optimize Cribl Search](#)
- [Create a Dashboard](#)
- [Edit a Dashboard](#)

5. CRIBL LAKE PERMISSIONS

Manage access to Cribl Lake by assigning product-level Permissions.

You can manage access to Lake Datasets by assigning specific product-level Permissions within each [Workspace](#) to [Members](#) and [Teams](#).

You can assign the following Permissions at the Cribl Lake product level.

Permission	Read Only	Editor	Admin
Read Lake Datasets			
Create and edit Lake Datasets			
Delete Lake Datasets			
Read Lakehouses			
Assign and unassign Lake Datasets to Lakehouses			

Some Cribl Lake capabilities require certain product-level Permissions set in other products from the Cribl Product Suite:

Capability	Permission
View the Integrated with column in the Dataset table, listing the Cribl Lake Collectors and Destinations that a Lake Dataset is connected with.	Read Only, Editor, or Admin Permission for Cribl Stream.
Search a Lake Dataset directly from the Dataset table, the Member/Team needs Cribl Search access .	User, Editor, or Admin Permission for Cribl Search, and Read Only or higher Permission for this Lake Dataset .

6. CRIBL LAKE COLLECTOR

Send your data from Cribl Lake to Cribl Stream, using the **Cribl Lake Collector**.

Requirements for Cribl Lake Collector

Cribl Lake Collector can gather data from both Cribl-managed [Cloud](#) and customer-managed [hybrid](#) Stream Worker Groups. Hybrid Worker Groups must be running version 4.8 or higher.

Configure a Cribl Lake Collector

1. On the top bar, select **Products**, and then select **Stream**.
2. Under **Worker Groups**, select a Worker Group.
3. On the **Worker Groups** submenu, select **Data**, then **Sources**.
4. In the **Sources** tiles, under **Collectors**, select **Cribl Lake**.
5. Select **Add Collector** to open the **New Collector** modal
6. In the Collector modal, configure the following under **Collector Settings**:
 - **Collector ID**: Unique ID for this Collector. For example: myLakeCollector.
 - **Lake dataset**: Lake Dataset to collect data from.
 - **Tags**: Optionally, add tags that you can use to filter and group Sources in Cribl Stream's **Manage Sources** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.
7. Optionally, configure any [Result](#) and [Advanced](#) settings outlined in the below sections.
8. Select **Save**, then **Commit & Deploy**.

 You can't use QuickConnect to configure Cribl Lake Collector Sources.

Result Settings

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

Event Breakers

In this section, you can apply event breaking rules to convert data streams to discrete events.

Event Breaker rulesets: A list of event breaking rulesets that will be applied, in order, to the input data stream. Defaults to System Default Rule.

Event Breaker buffer timeout: How long (in milliseconds) the Event Breaker will wait for new data to be sent to a specific channel, before flushing out the data stream, as-is, to the Routes. Minimum 10 ms, default 10000 (10 sec), maximum 43200000 (12 hours).

Fields

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

Name: Field name.

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

Result Routing

Send to Routes: Toggle on (default) if you want Cribl Stream to send events to normal routing and event processing. Toggle off to select a specific Pipeline/Destination combination. Toggling off exposes these two additional fields:

- **Pipeline:** Select a Pipeline to process results.
- **Destination:** Select a Destination to receive results.

Toggling on (default) exposes this field:

- **Pre-processing Pipeline:** Pipeline to process results before sending to Routes. Optional.

This field is always exposed:

- **Throttling:** Rate (in bytes per second) to throttle while writing to an output. Also takes values with multiple-byte units, such as KB, MB, or GB. (Example: 42 MB.) Default value of 0 indicates no throttling.



You might toggle **Send to Routes** off when configuring a Collector that will connect data from a specific Source to a specific Pipeline and Destination. This keeps the Collector's configuration self-contained and separate from Cribl Stream's routing table for live data – potentially simplifying the Routes structure.

Advanced Settings

Advanced Settings enable you to customize post-processing and administrative options.

Environment: If you're using GitOps, optionally use this field to specify a single Git branch on which to enable this configuration. If empty, the config will be enabled everywhere.

Time to live: How long to keep the job's artifacts on disk after job completion. This also affects how long a job is listed in **Job Inspector**. Defaults to 4h.

Remove Discover fields: List of fields to remove from the Discover results. This is useful when discovery returns sensitive fields that should not be exposed in the Jobs user interface. You can specify wildcards (such as `aws*`).

Resume job on boot: Toggle on to resume ad hoc collection jobs if Cribl Stream restarts during the jobs' execution.

Verify Data Flow

To verify that the Collector actually collects data, you can start a single run in the [Preview](#) mode.

1. Select your Collector's **Run** action.
 2. Make sure mode **Preview** is selected and accept other default settings.
 3. Confirm with **Run**.
 4. Look at the preview screen to check that data is being collected from Cribl Lake.
-

7. CRIBL LAKE DESTINATION

Send your data from Cribl Stream to Cribl Lake, using the **Cribl Lake Destination**.

The **Cribl Lake Destination** is a [Cribl Stream](#) feature that optimizes data management for two other Cribl products: It sends data to Cribl Lake, and automatically selects a partitioning scheme that works well with Cribl Search.



Type: Non-Streaming | TLS Support: Yes | PQ Support: No

Requirements for Cribl Lake Destination

The Cribl Lake Destination can send data to both Cribl-managed [Cloud](#) and customer-managed [hybrid](#) Stream Worker Groups. Hybrid Worker Groups must be running Cribl Stream version 4.8 or later.

On hosts for hybrid Worker Groups, this Destination requires outbound HTTP/S and TCP access to port 443. This Destination does not allow selecting a different port. For details, see [Troubleshoot Hybrid Access to Cribl Lake](#).

Prepare Data for Use with Lakehouse

Storing data in regular [Lake Datasets](#) does not require setting a schema. However, if you want to make full use of the [Lakehouse](#) functionality, you need to make sure that events sent from Cribl Stream are parsed into distinct fields before sending to Cribl Lake/Lakehouse.



To do this, use the Stream [Parser](#) Function to ensure that all events have named fields.

Configure a Cribl Lake Destination

1. On the top bar, select **Products**, and then select **Stream**.
2. Under **Worker Groups**, select a Worker Group. Next, you have two options:
 - To configure via the graphical [QuickConnect](#) UI:
 1. On the **Worker Groups** submenu, select **Routing**, then **QuickConnect**.
 2. Select **Add Destination** at right.
 3. Hover over **Cribl Lake** from the list of tiles.
 4. Select **Add New** to open a **New Destination** modal.

- Or, to configure via the [Routing](#) UI:

1. On the **Worker Groups** submenu, select **Data**, then **Destinations**.
2. In the **Destinations** tiles, select **Cribl Lake**.
3. Select **Add Destination** to open a **New Destination** modal.

3. In the Destination modal, configure the following under **General Settings**:

- **Output ID**: Enter a unique name to identify this Cribl Lake Destination.
- **Lake dataset**: Select a Lake Dataset to send data to.



You can't target the built-in `cribl_logs` and `cribl_metrics` Lake Datasets with this Destination.

4. Next, you can configure the following **Optional Settings** that you'll find across many Cribl Destinations:

- **Backpressure behavior**: Whether to block or drop events when all receivers are exerting backpressure. (Causes might include an accumulation of too many files needing to be closed.) Defaults to `Block`.
- **Tags**: Optionally, add tags that you can use to filter and group Destinations in Cribl Stream's **Manage Destinations** page. These tags aren't added to processed events. Use a tab or hard return between (arbitrary) tag names.

5. Optionally, configure any [Post-Processing](#) settings outlined in the below sections.

6. Select **Save**, then **Commit & Deploy**.

7. Verify that data is [searchable](#) in Cribl Lake.

Processing Settings

Post-Processing

Pipeline: [Pipeline](#) or [Pack](#) to process data before sending the data out using this output.

System fields: A list of fields to automatically add to events that use this output. By default, includes `cribl_pipe` (identifying the Cribl Stream Pipeline that processed the event). Supports `c*` wildcards. Other options include:

- `cribl_host` – Cribl Stream Node that processed the event.
- `cribl_input` – Cribl Stream Source that processed the event.
- `cribl_output` – Cribl Stream Destination that processed the event.
- `cribl_route` – Cribl Stream Route (or QuickConnect) that processed the event.
- `cribl_wp` – Cribl Stream Worker Process that processed the event.

Troubleshooting Cribl Lake Hybrid Access

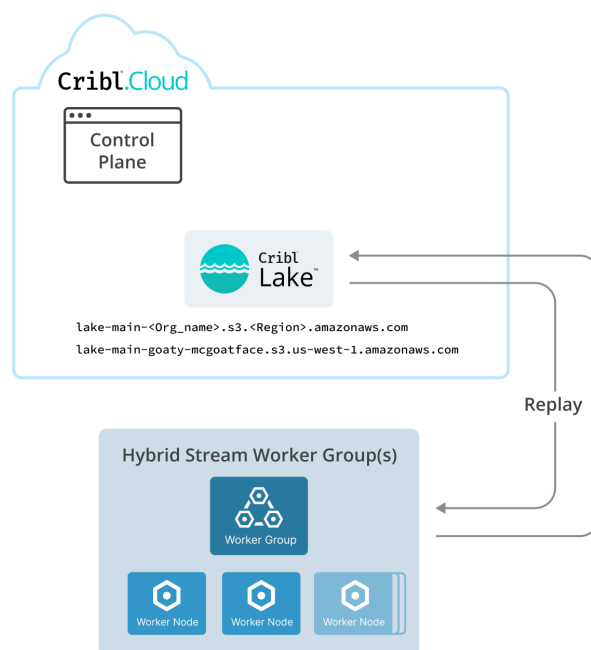
When running the Cribl Lake Destination on hybrid Worker Groups, you might receive SSL errors, or errors about being unable to connect to a host. This indicates that these Groups have restricted access to the internet through a firewall or proxy that performs SSL inspection. To reach the Cribl.Cloud Organization that hosts your Cribl Lake instance, you can remedy this as follows:

1. At the lower left of your Cribl.Cloud Organization, select **Organization Details**.
2. In the resulting fly-out, note the **Organization ID** and (AWS) **Region**.
3. Swap those two strings for the two placeholders in this fully qualified domain (FDQN) format:
`lake-main-<organizationId>.s3.<region>.amazonaws.com.`

With a fictional **Organization ID**, your FQDN might be:

`lake-main-goaty-mcgoatface.s3.us-west-1.amazonaws.com.`

4. Provide the resulting FQDN to your Security team, asking them to add a corresponding exception on the firewall or proxy.
5. Verify that outbound port 443 is open for TCP – this is **required** to access the FQDN.
6. Test the Cribl Lake Destination: Ensure that the target URL resolves, and that Cribl Stream can successfully send data to the corresponding Lake Dataset.



Firewall exception for hybrid output to Cribl Lake

8. CRIBL LAKE DIRECT ACCESS

Ship your data straight into Cribl Lake, bypassing Cribl Stream.

You can use the Direct Access option to archive certain data directly to Cribl Lake. This option bypasses routing inbound data through Cribl Stream. Once your data is integrated with Cribl Lake, Cribl Stream and Cribl Search can access it as a Lake Dataset.

Requirements for Direct Access

Direct Access is available for Splunk Cloud DDSS (Dynamic Data Self Storage) data. To set up this integration, you must create a new DDSS index.

Your Cribl.Cloud Organization must be in the same AWS Region as your Splunk Cloud Platform environment.

See also [Limitations](#) below.

Configure DDSS Direct Access

To configure this integration, you'll work back and forth between Splunk Cloud and Cribl Lake to share linking identifiers. The following subsections list the major steps, in sequential order. Keep your eyes on the shore!

Create Splunk Index and Storage

Start in your Splunk Cloud Platform, where you'll create a new index and corresponding storage details:

1. Select **Settings > Indexes > New Index**.
2. In the **Dynamic Data Storage** field, select **Self Storage**.
3. Select **Create a self storage location**.
4. On the resulting **Dynamic Data Self Storage** page, give your location a **Title** and (optionally) a **Description**.
5. Below the **Amazon S3 bucket name** field, select the **Splunk Cloud ID**. Copy it to your clipboard for the next section.

Create Cribl Lake Integration

In Cribl Lake, create the integration with your DDSS bucket:

1. From the sidebar, select **Direct Access**.
2. Accept the default **Splunk Cloud Self Storage** Source, by selecting **Next**.
3. In the **Splunk Cloud ID** field, paste the ID that you copied from Splunk Cloud in the previous section.

4. Optionally, enter a **Description** of this Lake integration's purpose.
5. From the **Region** drop-down, select the AWS Region of your Splunk Cloud Platform.
6. Select **Next**.
7. Copy the Cribl Lake bucket name to your clipboard for the next section.

The screenshot shows the 'New Source' configuration page in the Cribl Lake interface. The left sidebar contains navigation links: Lake, Products, Exciting Stucco Search Team, main, Collapse, Lake Home, Datasets, Lakehouses, Direct Access (highlighted with a 'Preview' badge), and Monitoring. The main content area is titled 'Direct Access / New Source' and features a progress bar with three steps: 1. Source, 2. Configuration (active), and 3. Bucket policy. A yellow warning box states: 'Your Cribl.Cloud region must be in the same AWS region as your Splunk Cloud Platform environment.' Below this, there are input fields for 'Splunk Cloud ID*' (with placeholder 'Enter Splunk Cloud ID'), 'Description' (with placeholder 'Enter description'), and 'Region*' (a dropdown menu with 'Select Splunk Cloud Region'). At the bottom are 'Previous' and 'Next' buttons.

Configuring Cribl Lake matching bucket

Connect Splunk Cloud to Your Lake Bucket

Next, return to your Splunk Cloud Platform, to point this environment to the bucket you've created in Cribl Lake:

1. In the **Amazon S3 bucket name** field, paste Cribl Lake bucket name you copied in the previous section.
2. Select **Generate** to generate a bucket policy.
3. Copy the resulting bucket policy to your clipboard for the next section.

Provision Your Lake

Return to Cribl Lake to link your bucket policy and provision the new Lake:

1. In the **AWS S3 bucket policy** pane, paste the Splunk Cloud bucket policy you copied in the previous section.
2. Select **Save**.
3. Wait a few minutes for your new Lake to be provisioned.

Direct Access / **New Source: Splunk Cloud Self Storage**

☒ Configuration
 2 Bucket policy

In Splunk Cloud Platform, create a new Self Storage location, and enter `bucket-captain@work-13731-jc074-295665257618-us-east-1` in the AWS S3 bucket name field. Generate the AWS S3 bucket policy and copy the output below:

AWS S3 bucket policy*

Enter AWS S3 bucket policy

Cribl Lake modal for bucket name and policy

Test and Submit the Connection

After provisioning is complete in Cribl Lake, test the connection from Splunk Cloud:

1. In the **Self Storage Locations** dialog, select **Test**.

Splunk Cloud will write a 0 KB test file to the root of your S3 bucket, to verify that Splunk Cloud Platform has permissions to write to the bucket.

2. If you see a success message, select **Submit** to finish linking your DDSS location to Cribl Lake.
3. If the test fails, see [Troubleshooting Direct Access](#).

Create Cribl Lake Dataset

Once you've verified the connection from Splunk Cloud, your final setup step is to create a Lake Dataset to contain your data. Follow the steps in [Create a New Lake Dataset](#), being careful to add these specific requirements to handle DDSS data:

- Give the Dataset an **ID** that exactly matches the Splunk index name you assigned in [Create Splunk Index and Storage](#) earlier on this page. (Otherwise, you will be unable to search or replay your data.)
- Set the **Data format** to DDSS.

Save your new Dataset, and your Splunk data should begin flowing into it. That's it!

Limitations of Direct Access

A Lake Dataset created and populated via Direct Access is read-only to other Cribl products. So, although both Cribl Stream and Cribl Search can process data from the Dataset, neither product can write to it.

A Lake Dataset created via Direct Access cannot be assigned to a [Lakehouse](#).

Breaking a Direct Access Connection

After you link Splunk and Cribl resources, their interdependence means that deleting resources on either side has the following consequences:

- In Cribl Lake, deleting a Splunk Cloud Self Storage Source means that you will lose all your Splunk data archived here. Also, Splunk Cloud will no longer be able to write to Cribl Lake through this integration. However, this deletion will not affect your data flow and storage within Splunk Cloud.
- In Splunk Cloud, deleting a Self Storage location that's linked to a Cribl Lake Dataset means that Splunk Cloud will no longer archive data to Cribl Lake. However, this deletion will not affect your data already stored within Cribl Lake.
- However, if you cancel your Splunk Cloud **account**, your Splunk Cloud Self Storage Source in Cribl Lake can still remain intact. This will allow Cribl Stream (Collector) and Cribl Search (queries) to continue reading your Dataset's current contents. However, no new data can be written to the Dataset.

Troubleshooting Direct Access

If Splunk Cloud fails to write the [test file](#) to your Lake Dataset, look for the following possible sources of error. Correcting each of these requires re-creating your integration from scratch:

- Regions mismatched between Splunk Cloud and Cribl.Cloud.
 - Bucket name does not start with the Splunk Cloud ID.
 - Amazon S3 bucket policy not [pasted in](#) accurately.
-

9. INTEGRATING CRIBL LAKE WITH CRIBL EDGE

Archive your Cribl Edge data to Cribl Lake, via Cribl Stream.

To send data from Cribl Edge to Cribl Lake, you need to route it through Cribl Stream, using the Cribl HTTP or Cribl TCP Destination/Source pair. Either option prevents double billing when you ingest the data from Edge to Stream.

The main steps are:

1. [Send data](#) from Cribl Edge to a Stream Worker Group.
2. [Receive data](#) in Cribl Stream.
3. [Send data](#) from Cribl Stream to Cribl Lake.



To learn more about communicating between Cribl Edge and Cribl Stream, see the [Cribl Edge to Cribl Stream](#) guide.

The [Edge Getting Started tutorial](#) presents a similar use case with detailed steps for configuring the Cribl HTTP Sources and Destination to seamlessly send data between Edge and Stream.

Send Data from Cribl Edge

1. In Cribl Edge, create a new [Cribl HTTP](#) Destination. (You could also use [Cribl TCP](#), but this example will use the former.)
2. Next, use QuickConnect to pass your Edge data to this Destination.

Receive Data in Cribl Stream

1. In Cribl Stream, create a new [Cribl HTTP](#) Source.
2. Make sure that the **Address** and **Port** fields in the Source's configuration correspond to what you configured in Cribl Edge.

Send Data to Cribl Lake

1. Once data is flowing from Cribl Edge to Cribl Stream, create a [Cribl Lake Destination](#) in the same Worker Group.
2. Select the Lake Dataset you want to use as the target for the Destination.
3. Finally, connect the Cribl HTTP Source with the new Cribl Lake Destination via QuickConnect.

Verify Data Flow

To make sure that data is flowing correctly into Cribl Lake, you can run a test [search](#) over the selected Lake Dataset.

10. KNOWN ISSUES

View filterable Known Issues for [Cribl Lake](#). Check if issues are resolved or find temporary workarounds.
